FORMAL SPECIFICATION AND MODEL CHECKING OF DISTRIBUTED SYSTEMS WITH REWRITING LOGIC META-PROGRAMMING FACILITIES

DOAN, HA THI THU

March, 2019

Abstract

In recent decades, key software systems on which human beings heavily rely on are in the form of distributed systems. Such systems are quite complex and thus very hard to design and verify. Model checking is a popular automatic formal technique and highly suitable for verifying distributed systems. Several researchers have attempted to formally analyze and verify distributed systems. However, these are several tough problems or new forms of distributed systems that have not been tackled well by existing approaches (or tools).

Control algorithms are a large important class of distributed algorithms, which deal with significant problems, such as snapshot recording algorithms and checkpointing algorithms. One main characteristic of these algorithms is that they are superimposed on underlying distributed systems. Although some research have been conducted to formally verify control algorithms, all of them directly require humans to specify by hand underlying distributed systems on which control algorithms are superimposed. It is expected to have more general approaches. However, it is challenging to specify control algorithms in almost all existing specification languages for model checkers because it is necessary to treat an underlying distributed system as the data that is handled by these algorithms. Therefore, it is one of the challenging problems to be solved that how to specify control algorithms but not underlying distributed systems on which control algorithms are superimposed is.

Recent advances in distributed computing highlight models and algorithms for autonomous mobile robots that self-organize and cooperate together in order to solve a global objective. Due to the mobility aspect, robot algorithms are often complex, arguably even more complex than classical distributed systems. Designing and analyzing mobile robot algorithms is notoriously difficult.

Rewriting logic is a natural model of computations for concurrency and communication systems. Several specification languages based on rewriting logic, such as Maude, CafeOBJ and Elan have been designed and implemented. Moreover, rewriting logic is a reflective logic that can be faithfully interpreted in itself. Rewriting logic is highly suitable to formalyze distributed algorithms. However, rewriting logic only

at the object level may not be powerful enough to precisely specify some distributed algorithms.

This thesis focuses on exploiting Rewriting logic meta-programming facilities to formalize the distributed algorithms that have not been tackled well by existing approaches (or tools), as well as by any methods (or tools) based on rewriting logic at the object level. The aim of the research is to achieve how to tackle two important families of distributed systems, namely control algorithms and mobile robot algorithms, with rewriting logic meta-programming facilities. Theoretically, we have faced the above mentioned problems by moving from the object level to the meta level, namely that it is necessary to deal with the specifications of underlying distributed systems as data for the specification of a control algorithm and the succinct specification of mobile robot algorithms as data for a translator by which the succinct ones are transformed to those that can be directly treated by Maude.

First of all, we propose a new approach to specifying and model checking control algorithms. Meta-programming technique is applied to the challenges above-mentioned. We have used meta-programs as formal specifications of control algorithms. A control algorithm is specified as a meta-program that takes the specification of an underlying distributed system as an input and generates the specification of the underlying distributed system on which the control algorithm is superimposed (UDS-CA). A control algorithm is only specified at once, and for each underlying distributed system, the specification of the UDS-CA is automatically obtained. Furthermore, we propose a technique that takes the number of each kind of entities used, generate all possible initial states that satisfy some constraints and conduct model checking experiments for all the initial states, which makes it more likely to detect a subtle flaw lurking in a control algorithm or improves the confidence in the correctness of a control algorithm. We have conducted two case studies, which specify and model check snapshot and checkpointing algorithms.

Several classic distributed algorithms have been formally verified with some techniques based on rewriting logic. We aim to obtain similar achievements for distributed mobile robot systems - a new form of distributed systems. We come up with formal specification and model checking based on rewriting logic for mobile robot algorithms. We have conducted two case studies in which we specify and model check an exploration algorithm and a gathering algorithm on rings in Maude. However, no existing specification language is designed for mobile robot algorithms on rings: rings are not directly supported by such languages and specifications of such algorithms are far from the corresponding mathematical descriptions. This is because of the particular symmetries owned by *rings*. Consequently, we need to specify rings by adapting other defined structures, such as *sets* and *sequences*. It, therefore, makes the specification task tedious as well as time-consuming, while the specifications obtained are complicated and lengthy. It is worth providing a specification environment in which rings are directly supported. An environment and a domain-specific language (DSL) for specifying and model checking mobile robot algorithms on rings (or mobile ring robot algorithms) are proposed. First, we develop Maude Ring Specification Environment (Maude RSE), a ring specification environment that explicitly supports ring-shaped networks. Then, we build our DSL, Mobile Ring Robot Maude (MR^2-Maude), on top of Maude RSE. MR^2-Maude makes it possible to specify mobile ring robot algorithms in such a way that the specifications are as close as possible to their mathematical descriptions. One key

underlying these tools is pattern matching between ring patterns and ring instances, called "ring pattern matching." The advantages of Maude RSE and MR^2-Maude are demonstrated by case studies analyzing exploration and gathering mobile robot algorithms.

Keywords: control algorithm, mobile robot algorithm, model checking, meta-programming, domain-specific language.