JAIST Repository
https://dspace.jaist.ac.jp/

Title	Revisiting Model Checking of Chandy-Lamport Distributed Snapshot Algorithm
Author(s)	Doan, Thi Thu Ha
Citation	
Issue Date	2015-09
Type	Thesis or Dissertation
Text version	author
URL	http://hdl.handle.net/10119/12928
Rights	
Description	Supervisor:Kazuhiro Ogata, 情報科学研究科, 修士

JAIST
JAPAN
ADVANCED INSTITUTE OF
SCIENCE AND TECHNOLOGY

Japan Advanced Institute of Science and Technology

Revisited Model Checking of Chandy-Lamport Distributed Snapshot Algorithm

DOAN, Ha Thi Thu (1310210)

School of Information Science,
Japan Advanced Institute of Science and Technology

August 6, 2015

Keywords: distributed snapshot algorithm, reachability, state machine, model checking.

In recent decades, many applications have relied on distributed systems, which involve multiple processes connected by channels. Due to the fact that distributed systems should be fault tolerant because they need to run for a long time, keeping on providing services to human beings, other systems, etc. To make distributed systems fault tolerant, it is necessary to use many non-trivial distributed algorithms, such as snapshot algorithms and checkpointing algorithms. Since many problems on distributed systems, such as recovering from faulty states and detecting stable properties can be cast to taking global snapshots of distributed systems, distributed snapshot algorithms are crucial.

A distributed system consists of a finite set of processes and channels. The processes communicate by sending and receiving messages through channels. A global state of a distributed system includes the states of all processes and all channels in the system, where the state of a channel is a sequence of messages sent along the channel, but not been received. Distributed snapshot algorithms help to determine global states of a distributed system (called global snapshots) during a computation. One desired property distributed snapshot algorithms should enjoy is as follows: let s_1 be the state in which a distributed snapshot algorithm concerned initiates, s_2 be the state in which the distributed snapshot algorithm terminates, and s_* be the snapshot taken, and then s_* is reachable from s_1 and s_2 is reachable from s_*, whenever the distributed snapshot algorithm terminates. The property is called the distributed snapshot reachability ($\mathcal{DSR}$) property. Note that the snapshot may not appear in the actual computation from the start state to the finish state.

Taking global snapshots of a distributed system is not straightforward because distributed systems are asynchronous and processes do not share clocks and memory. These lead to that all processes cannot record their local states at exactly the same time. What we obtained might be inconsistent global states. Therefore, Chandy and Lamport have proposed a distributed snapshot algorithm called $\mathcal{CLDSA}$ in this thesis, by which processes can record their own states and the states of communication channels such that the combination of process states and channel states form a consistent global state.

With the advancement of computer and model checking technologies, many model checkers, such as Spin and NuSMV have been developed and applied to formal verification of

Copyright © 2015 by DOAN, Ha Thi Thu

various kinds of software and hardware systems. To the best of our knowledge, however, application of model checking to formal verification of distributed snapshot algorithms have not been fully investigated. Since distributed snapshot algorithms are non-trivial, the problem to model check that the distributed snapshot algorithms enjoys the $\mathcal{DSR}$ property is not straightforward. There is an existing study in which Maude has been used to model check that $\mathcal{CLDSA}$ enjoys the $\mathcal{DSR}$ property. Maude is a specification and programming language system based on rewriting logic, and equipped with model checking facilities: the LTL model checker and the search command. The search command is used in the existing study. The authors of the existing study, however, do not discuss whether their definition of the $\mathcal{DSR}$ property faithfully expresses the property. In addition, we recognize that the informal description of the $\mathcal{DSR}$ property involves both an underlying distributed system ($\mathcal{UDS}$) and the $\mathcal{UDS}$ superimposed by $\mathcal{CLDSA}$, while the $\mathcal{DSR}$ property encoded in terms of the Maude search command involves only the $\mathcal{UDS}$ superimposed by $\mathcal{CLDSA}$ in the existing study. Consequently, we do not think that the existing study provides the good foundation to guarantee that the $\mathcal{DSR}$ property has been sufficiently model checked.

Therefore, it is necessary to faithfully express the $\mathcal{DSR}$ property, and then consider the similarities between the new formal definition and the existing one. Moreover, one significant characteristic of distributed snapshot algorithms is that distributed snapshot algorithms should run concurrently with but not alter the $\mathcal{UDS}$. Since this property relate to computation and the behaviors of a $\mathcal{UDS}$, however, it is not straightforward to prove that a distributed snapshot algorithm satisfies this property manually. Chandy and Lamport in the original paper asserts that $\mathcal{CLDSA}$ satisfy this property, but they have not yet proved it. Until now, we have not yet found out any reliable studies, in which this property is proven. In addition, this property is not mentioned in the existing study. But, we should prove that $\mathcal{CLDSA}$ does not alter the behaviors of a $\mathcal{UDS}$.

The purpose of our research is to revisit model checking of $\mathcal{CLDSA}$. Carefully investigating the informal description of the $\mathcal{DSR}$ property, there is another existing study on finding the way to faithfully express the $\mathcal{DSR}$ property. However, the study has not yet completed. The technical report has not been reviewed. Our research complete this existing study giving a formal definition of the $\mathcal{DSR}$ property, which is more likely to faithfully express the informal description. Then we prove that the new definition of the $\mathcal{DSR}$ property is equivalent to the definition in the existing study to confirm the validity of the model checking approach used in the existing study. Since the new definition involves both a $\mathcal{UDS}$ and the $\mathcal{UDS}$ superimposed by $\mathcal{CLDSA}$, it is not straightforward to directly model check the new definition with any existing model checker. Hence the equality of the two definitions also asserts that we can use the model checking approach used in the existing study to model check the new definition. Moreover, we prove that $\mathcal{CLDSA}$ does not alter the behaviors of a $\mathcal{UDS}$.

Since state machines are suitable to formalize concurrent systems including distributed systems, state machines are used in our research to formalize a $\mathcal{UDS}$ and the $\mathcal{UDS}$ superimposed by $\mathcal{CLDSA}$. To faithfully express the informal description of the $\mathcal{DSR}$ property, we first formalize a $\mathcal{UDS}$ and the $\mathcal{UDS}$ superimposed by $\mathcal{CLDSA}$ as state machines $M_{\mathcal{UDS}}$ and $M_{\mathcal{CLDSA}}$, respectively. The $\mathcal{DSR}$ property is formalized based on them. Maude notation is used to describe state machines. Then we prove Theorem 1

saying that our new definition is equivalent to the definition of the $\mathcal{DSR}$ property in the existing study. To prove the Theorem 1, we prove Proposition 1 and Lemma 1. Lemma 1 asserts that reachability is preserved between $M_{\mathcal{UDS}}$ and $M_{\mathcal{CLDSA}}$. Proposition 1 says that whenever $\mathcal{CLDSA}$ terminates in state s, there is no marker in the start state, the snapshot and the finish state. We prove as Lemma 2 and Lemma 3 that one-step reachability is preserved between $M_{\mathcal{UDS}}$ and $M_{\mathcal{CLDSA}}$ to prove Lemma 1. The proof of Theorem 1 follows from Proposition 1 and Lemma 1. Furthermore, a binary simulation between $M_{\mathcal{UDS}}$ and $M_{\mathcal{CLDSA}}$ is used to prove that $\mathcal{CLDSA}$ does not alter the behaviors of a $\mathcal{UDS}$. Since $\mathcal{CLDSA}$ works by using a special message called marker, $\mathcal{CLDSA}$ does not alter the behaviors of a $\mathcal{UDS}$ means that excepting for putting markers in a $\mathcal{UDS}$, the algorithm does not change the state of all processes and channels. We propose the binary relation $\mathbf{r}$ between $M_{\mathcal{UDS}}$ and $M_{\mathcal{CLDSA}}$ saying that for each $s1 \in S_{\mathcal{UDS}}$ and each $s2 \in S_{\mathcal{CLDSA}}$, $\mathbf{r}(s1, s2)$ if and only if $s1$ is the same as the state obtained by deleting all markers from $s2$. To guarantee that $\mathcal{CLDSA}$ does not alter the behaviors of a $\mathcal{UDS}$, we prove Theorem 2 saying that $\mathbf{r}$ is a bi-simulation relation between $M_{\mathcal{UDS}}$ and $M_{\mathcal{CLDSA}}$.

Summarizing the research results, we have given a more faithful formal definition of the $\mathcal{DSR}$ property. Moreover, we have proved Theorem 1 saying that our formalization of the $\mathcal{DSR}$ property is equivalent to the existing one for each $M_{\mathcal{UDS}}$. Theorem 1 guarantees that it suffices to model check the definition used in the existing study for $\mathcal{CLDSA}$ and the existing model checking approach can be used for this end. We have also proved Theorem 2 asserting that $M_{\mathcal{CLDSA}}$ simulates $M_{\mathcal{UDS}}$ and vice versa to guarantee that $\mathcal{CLDSA}$ does not alter the behaviors of the $\mathcal{UDS}$.