Model-Checking Observational Transition Systems with Maude

Weiqiang Kong[1], Kazuhiro Ogata[1, 2], and Kokichi Futatsugi[1]
[1]Japan Advanced Institute of Science and Technology (JAIST)
1-1 Asahidai, Nomi, Ishikawa 923-1211, Japan
Tel: +81-90-3763-9683, Fax: +81-761-51-1370
[2] NEC Software Hokuriku, Ltd.
1 Anyoji, Tsurugi, Ishikawa 920-2141, Japan
Fax: +81-761-51-1370
E-mail: weiqiang@jaist.ac.jp, ogatak@acm.org, kokichi@jaist.ac.jp

Abstract: We describe how to model-check observational transition systems, or OTSs with Maude. Maude is a specification and programming language based on rewriting logic. We describe how to write OTSs in Maude and how to model-check them with the Maude LTL model-checker. We demonstrate our method using a case study – model-checking a workflow system that deals with travel expense reimbursement. The main goal of our analysis is to check whether the existence of security consideration will prevent completion of the execution of the workflow.

1. Introduction

We have been successfully applying the OTS/CafeOBJ [1] method to modeling, specification and verification of distributed systems such as security protocols [2, 3]. In the OTS/CafeOBJ method, systems are modeled as observational transition systems, or OTSs, which are the definition of transition systems for writing them in terms of equations; and OTSs are then written in CafeOBJ [4], an algebraic specification language. We verify that the OTSs have certain properties by writing proof scores in CafeOBJ and executing the proof scores with CafeOBJ system.

In the OTS/CafeOBJ method, theorem proving technique is used to verify that OTSs have properties. There is also another important formal verification method – model-checking technique [5], which can be used to verify that (usually finite) systems have properties. Although theorem proving technique would not be totally replaced by model-checking technique, model-checking technique is generally easier to use than theorem proving technique and can be used as the complement of theorem proving technique. You may find errors lurked in your specification by model-checking the specification before you begin to verify the specification using theorem proving technique.

In this paper, we describe how to model-check OTSs with Maude [6]. As a sibling language of CafeOBJ, Maude is a specification and programming language based on rewriting logic, which has model-checking facilities. We describe how to write OTSs in Maude and how to model-check them with the Maude LTL model-checker.

2. Observational Transition Systems

Observational transition systems, or OTSs are the definition of transition systems for writing transition systems in terms of equations. We assume that there exists a universal state space called Υ. We also suppose that each data type used has been defined beforehand. An OTS S consists of $\langle \mathcal{O}, \mathcal{I}, \mathcal{T} \rangle$, where $\mathcal{O}$ is a set of observers that observe the state changes of S, $\mathcal{I}$ is the set of initial states of S, and $\mathcal{T}$ is a set of conditional transition rules that change the states of S.

Multiple similar observers and transition rules may be indexed. Generally, observers and transition rules are denoted by $o_{i1,\ldots,im}$, and $t_{j1,\ldots,jn}$, repectively, provided that m, n ≥ 0 and we assume that there exist data types D_k such that k $\in D_k$(k $= i_1,\ldots,i_m, j_1,\ldots,j_n$). For example, an integer array a possessed by a process p may be denoted by an observer a_p, and the increment of the ith element of the array may be denoted by a transition rule $inc\text{-}a_{p,i}$.

3. Description of OTSs in Maude

Maude is a specification and programming language based on rewriting logic. We describe how to write an OTS S in Maude. $\mathcal{O}$ and $\mathcal{T}$ are denoted by sorts, say **OValue** and **TRule**, repectively. The state space Υ is denoted by a sort, say **Sys**. A snapshot of S is represented by a multiset, or a bag of observers and transition rules. **OValue** and **TRule** are then declared as subsorts of **Sys**.

Generally, a snapshot of S is as the following form:

ovalue-1…ovalue-M trule-1…trule-N

where *ovalue-i* ($i = 1,…,M$) is a term denoting an observer, and *trule-i* ($i = 1,…,N$) is a term denoting a transition rule.

An observer $o_{i1,\ldots,im} \in \mathcal{O}$ is denoted by and operator. We assume that the data types D_k (k $= i_1,\ldots,i_m$) and D are defined and there exist sorts V_k (k $= i_1,\ldots,i_m$) and V corresponding to the data types. The operator denoting $o_{i1,\ldots,im}$ is declared as follows:

op (o[_,…,_] : _) : $V_{i1} \ldots V_{im}$ V -> OValue .

A transition rule $t_{j1,\ldots,jn} \in \mathcal{T}$ is denoted by an operator. We assume that the data types D_k (k $= i_1,\ldots,i_m$) and D are defined and there exist sorts V_k (k $= i_1,\ldots,i_m$) and V corresponding to the data types. The operator denoting $t_{j1,\ldots,jn}$ is declared as follows:

op r : $V_{j1} \ldots V_{jn}$ -> TRule .

Transition rules are defined using Maude rules. We suppose that observers needed and affected by the execution of the transition rule $t_{j_1,\ldots,j_n}$ are $o^1_{i^1_1,\ldots,i^1_{m_1}},\ldots,o^l_{i^l_1,\ldots,i^l_{m_l}}$, which are supposed to be denoted by operators (o^1[_,…,_] : _) ,…, (o^l[_,…,_] : _). Then the transition rule $t_{j_1,\ldots,j_n}$ detnoted by r is generally defined as follows:

$$
\begin{array}{l}
\texttt{crl[rule-}r\texttt{] :} \\
\quad r(X_{j_1},\ldots,X_{j_n}) \\
\quad (o^1[X_{i^1_1},\ldots,X_{i^1_{m_1}}] : X_1)\ldots(o^l[X_{i^l_1},\ldots,X_{i^l_{m_l}}] : X_l) \\
\quad \texttt{=>} \\
\quad r(X_{j_1},\ldots,X_{j_n}) \\
\quad (o^1[X_{i^1_1},\ldots,X_{i^1_{m_1}}] : X'_1)\ldots(o^l[X_{i^l_1},\ldots,X_{i^l_{m_l}}] : X'_l) \\
\quad \texttt{if } c\text{-}r(X_{j_1},\ldots,X_{j_n},X_{i^1_1},\ldots,X_{i^1_{m_1}},X_1,\ldots,X_{i^l_1}, \\
\qquad\qquad \ldots,X_{i^l_{m_l}},X_l)\ \ .
\end{array}
$$

where c-r is the operator denoting $c_{t_{j_1,\ldots,j_n}}$. rule-r is the label of the rule, which is optional. $X_k(k = j_1,\ldots,j_n,i^1_1,\ldots,i^1_{m_1},1,\ldots,i^l_1,\ldots,i^l_{m_l},l)$ is a term or a variable for the intended sort. X_k' (k = 1,…,l) denotes the value returned by observer $o^k_{i^k_1,\ldots,i^k_{m_k}}$ in the successor state with repspect to $t_{j_1,\ldots,j_n}$.

4. Model-Checking OTSs

We now briefly describe how to model-check that an OTS has properties with Maude. We suppose that the OTS is written in Maude as a module whose name is SYSTEM. Model-checking OTSs usually consists of the following procedures:

1. Define state predicates with which properties are described. Such state predicates are declared in a module, say SYSTEM-PREDS. Inside this module, linear temporal logic (LTL) is described (actually imported), and a satisfaction operator |= is declared that is used to define state predicates. A state predicate denoted by a term *pred* holds in a state denoted by *state* is defined as follows: **eq *state* |= *pred* = true .** .
2. Define propositional LTL formulas denoting properties to be checked for the OTS and also initial states of the OTS. Such formulas and initial states are described in a module, say SYSTEM-CHECK. Propositional LTL formulas are constructed of state predicates declared in SYSTEM-PREDS, Boolean connectives and temporal operators.
3. Model-check that all the states reachable from the initial state satisfies the propositional LTL formula as follows: **red modelCheck(*init*, *prop*) .** , where **red** is a Maude command, *init* is a term denoting an initial state and *prop* is a term denoting a propositional LTL formula to be checked.

5. Case Study and Conclusion

We demonstrate our method using a case study – model-checking a workflow system that deals with travel expense reimbursement. This workflow system has been broadly used to demonstrate workflow security considerations, such as Role-based Access Control (RBAC) and Separation of Duty (SoD) constraints.

RBAC is a natural mechanism to lighten the complexity of security administration, with the basic notion that permissions are associated with roles and users are assigned to appropriate roles. However, to satisfy the complex security policies of workflow systems, SoD constraints are also necessary, which aim at preventing fraud within an organization by not allowing any individual to have sufficient authority to perpetrate a fraud on his own [7].

Following our method, the workflow system, together with its security considerations (RBAC and SoD constraints), is modeled as an OTS. Activities, such as grant/revoke privilege to/from roles, are triggered by events that are generated by transition rules; and SoD constraints are specified in effective conditions that attached to each transition rule. The OTS is then written in Maude. We express safety and liveness properties of the workflow system in linear temporal logic (LTL) formulas, and verify that the OTS satisfies these properties by the Maude LTL model-checker. The main purpose of the verification is to check: whether the existence of security considerations will prevent completion of the execution of certain workflow instance, which is reflected by the liveness property.

Conclusion: The method described in this paper would be the first step toward translation of OTS/CafeOBJ specifications to Maude ones to model-check the OTS/CafeOBJ specifications [8].

References

[1] Kazuhiro Ogata and Kokichi Futatsugi, "Proof scores in the OTS/CafeOBJ method", In *FMOODS 2003*, Vol. 2884 of LNCS, pp. 170-184, Springer, 2003.

[2] Kazuhiro Ogata and Kokichi Futatsugi, "Rewriting based verification of authentication protocols", In *WRLA 2002*, Vol. 71 of ENTCS, Elsevier Science Publishers, 2002.

[3] Kazuhiro Ogata and Kokichi Futatsugi, "Formal analysis of the *i*KP electronic payment protocols", In *ISSS 2002*, Vol. 2609 of LNCS, pp. 441-460, Springer, 2003.

[4] Razvan Diaconescu and Kokichi Futatsugi, "CafeOBJ report", Vol. 6 of *AMAST Series in Computing*, World Scientific, Singapore, 1998.

[5] Jr. Edmund M. Clarke, Orna Grumberg, and Doron A. Peled, "Model Checking", The MIT Press, 2001.

[6] Manuel Clavel, Francisco Duran, Steven Eker, Patrick Lincoln, Narciso Marti-Oliet, Jose Meseguer, and Carolyn Talcott, "Maude 2.0 manual: Version 1.0", June 2003.

[7] Elisa Bertino, Elena Ferrari, and Vijay Atluri, "The specification and enforcement of authorization constraints in workflow management systems", *ACM Transactions on Information and System Security*, 2:65-104, 1999.

[8] Weiqiang Kong, Kazuhiro Ogata, Takahiro Seino, and Kokichi Futatsugi, "A toolkit for model-checking OTS/CafeOBJ specification with Maude", submitted, 2005.