Symbolic Approaches to Feature Interaction Detection

Tatsuhiro Tsuchiya
Osaka University
t-tutiya@ist.osaka-u.ac.jp

Masahide Nakamura
Nara Institute of Science & Technology
masa-n@is.aist-nara.ac.jp

Tohru Kikuno
Osaka University
kikuno@ist.osaka-u.ac.jp

Abstract

Feature interaction is a kind of inconsistent conflict between multiple communication services and considered an obstacle to developing reliable telephony systems. In this note we outline the results of applying symbolic model checking techniques to detection of feature interactions in service specifications.

1. Introduction

Feature interaction refers to situations where a combination of different services behaves differently than expected from the single services' behaviors. For example, consider a situation where user A has subscribed to the service *Originating Call Screening* (OCS) and does not want calls to user C to be put through, and user B has activated the service *Call Forwarding* (CF) to user C. In this situation, if A calls B, the intention of OCS not to be connected to C is invalidated since the call is put through to C by way of B. In today's intelligent telecommunication networks, feature interaction is considered a major obstacle to the introduction of new features and the provision of reliable services.

Much research has been conducted to tackle this problem [2]. In this note, we outline the preliminary results of applying *symbolic model checking* to detection of feature interactions in communication service specifications.

Model checking is a formal approach for verifying systems that are modeled as a finite state machine. For realistic designs, the number of states of the system can be very large and the explicit traversal of the state space may become infeasible. *Symbolic model checking* is one approach to this state explosion problem. This method alleviates the problem by implicitly representing the state space by boolean functions.

For any finite-state system, a state S can be represented as a boolean vector of fixed length. Any set A of states can be represented as a boolean function such that

$$f_A(S) = 1 \text{ iff } S \in A$$

Supported by a grant from Japanese Ministry of Education (No. 13224060)

The transition relation can be similarly encoded as follows.

$$T(S, S') = 1 \text{ iff there is a transition from } S \text{ to } S'$$

Forward state traversal can be performed by repeatedly computing

$$\exists S.(f_A(S) \wedge T(S, S'))|_{S' \to S}$$

The above formula represents the set of states that can be reached exactly in one step from the states in A. Backward state traversal can be carried out in a similar way.

Many symbolic model checking tools use *Binary Decision Diagrams* (BDDs) as the data structure to manipulate boolean functions, since boolean functions can often be represented by BDDs very compactly.

We developed a program that produces an input program of SMV, a well-know BDD-based model checker, from a given service specification and applied SMV to feature interaction detection. The results were quite encouraging. We had already developed a tool called SVAL [3], which uses explicit state representation in combination of symmetry reduction. For many cases, the time required by SMV for verification was less than half than SVAL.

To further improve the verification performance, we have been attempting to develop other techniques. In the rest of this note, we outline these techniques.

2. Model

We use *State Transition Rules* (STR) to model telecommunication services. Figure 1 shows the STR specification for the *Plain Old Telephone Service* (POTS). The rules are of the following form.

$$rule : \textit{pre-condition } [event] \textit{ post-condition.}$$

For example, consider two users A, B. Then the application of $r1$ to the initial state $\{idle(A), idle(B)\}$ yields either state $\{idle(A), dialtone(B)\}$ or state $\{idle(B), dialtone(A)\}$. Additional services can be described by adding rules to the POTS specification. Services we consider include: *Call Forwarding* (CF), *Originating Call Screening* (OCS), *Terminating Call Screening* (TCS), *Denied Origination* (DO), *Denied Termination* (DT), and *Direct Connect* (DC).

$r1 : idle(x)\ [offhook(x)]\ dialtone(x).$
$r2 : dialtone(x)\ [onhook(x)]\ idle(x).$
$r3 : dialtone(x),\ idle(y)\ [dial(x,y)]\ calling(x,y).$
$r4 : dialtone(x),\ \neg idle(y)\ [dial(x,y)]\ busytone(x).$
$r5 : calling(x,y)\ [onhook(x)]\ idle(x),\ idle(y).$
$r6 : calling(x,y)\ [offhook(y)]\ path(x,y),\ path(y,x).$
$r7 : path(x,y),\ path(y,x)\ [onhook(x)]\ idle(x),\ busytone(y).$
$r8 : busytone(x)\ [onhook(x)]\ idle(x).$
$r9 : dialtone(x)\ [dial(x,x)]\ busytone(x).$

Figure 1. STR specification for POTS.

3. Exploiting Symmetry

To further improve the performance of the BDD-based method, we have been attempting to exploit symmetry to reduce the state space. In telecommunication systems, it is usual that all subscribers of a service are guaranteed to be able to use the same functionality of the service. This symmetry in terms of users defines an equivalence relation on the states. For example, states $\{idle(A), dialtone(B)\}$ and $\{idle(B), dialtone(A)\}$ are in the same equivalence class.

The basic idea in exploiting symmetry is to check only the representative state of each equivalence class. Let $F(S, S')$ be the boolean function such that $F(S, S') = 1$ iff S' is the representative of the equivalence class that S belongs to. Suppose that $f_A(S)$ represents the set of representative states that have already been explored. Forward state traversal can then be performed by iterating the following two steps.

Step 1 : $f_B(S) := \exists S.(f_A(S) \wedge T(S, S'))|_{S' \to S}$
Step 2 : $f_A(S) := f_A(S) \vee \exists S.(f_B(S) \wedge F(S, S'))|_{S' \to S}$

In words, Step 2 maps newly visited states to their representatives.

In our experiment, we found that the method works best when the number of users is four. In this case, the verification time was reduced to almost half for many combinations of services. On the other hand, the number of users exceeded four, ordinary symbolic state traversal worked much better. This is because computing $F(S, S')$ consumes a large amount of time and space, thus diminishing the benefits of state space reduction. Currently we are trying to improve this technique to achieve further efficiency.

4. Bounded Model Checking

Bounded model checking has received recent attention as an efficient verification method [1]. The basic idea of this method is to reduce the model checking problem to the propositional satisfiability decision problem. For example, suppose that $I(S)$ and $FI(S)$ represent the initial state and the set of undesirable states where feature interaction occurs. Then the undesirable states can be reached within k steps from the initial state iff $I(S_0) \wedge T(S_0, S_1) \wedge T(S_1, S_2) \cdots \wedge T(S_{k-1}, S_k) \wedge (FI(S_0) \vee \cdots \vee FI(S_k))$ is satisfiable.

Table 1. Times required for verification (in seconds).

Services	SVAL	BC
CF+DT	2.04	0.17
CF+OCS	7.64	0.19
CF+TCS	0.06	0.18
DC+DT	24.75	0.05
DC+DO	0.19	0.03
DC+OCS	0.04	0.26
DC+TCS	0.04	0.28
DT+OCS	0.67	0.12
DT+TCS	0.02	0.01
OCS+TCS	0.02	0.11

For asynchronous systems, however, this method does not work well because the above formula tends to become very large for such systems. Because of the asynchronous nature of telecommunication systems, it is not practical to apply the original method to feature interaction detection.

Recently we developed an alternative method for 1-bounded Petri nets which uses a very succinct formula [4]. We also applied the method to feature interaction detection. Table 1 compares the time required to detect the first feature interaction between the bounded checking approach (denoted by BC) and SVAL. (We used a LINUX PC with a 850 MHz Pentium III processor.) Although the two methods exhibited similar performance, bounded model checking achieved significant reduction for some cases.

References

[1] A. Biere, A. Cimatti, E. M. Clarke, and Y. Zhu. Symbolic Model Checking without BDDs. In *Proceedings of Tools and Algorithms for the Analysis and Construction of Systems (TACAS'99), number 1579 in LNCS*, pages 193–207, 1999.

[2] D. O. Keck and P. J. Kuehn. The feature and service interaction problem in telecommunicatons systems: A survey. *IEEE Transactions on Software Engineering*, 24(10):779–796, October 1998.

[3] M. Nakamura and T. Kikuno. Feature interaction detection using permutation symmetry. In *Proc. of Fifth Int'l. Workshop on Feature Interactions in Telecommunication Networks and Distributed Systems (FIW'98)*, pages 193–207, 1998.

[4] T. Tsuchiya, E. Ashida, and T. Kikuno. A Simple and Efficient Bounded Model Checking Method for Asynchronous Systems. *(submitted for publication)*, 2002.