IEICE TRANS. INF. & SYST., VOL.E82–D, NO.10 OCTOBER 1999

PAPER

Exploiting Symmetric Relation for Efficient Feature Interaction Detection

Masahide NAKAMURA[†] *and* **Tohru KIKUNO**[†], *Members*

SUMMARY Feature interaction detection determines whether interactions occur or not between the new and existing telecommunication services. Most of conventional detection methods on state transition model utilize an exhaustive search. The exhaustive search is fundamentally very powerful in the sense that all interactions are exactly detected. However, it may suffer from the state explosion problem due to the exponential growth of the number of states in the model when the number of users and the number of features increase. In order to cope with this problem, we propose a new detection method using a state reduction technique. By means of a symmetric relation, called permutation symmetry, we succeed in reducing the size of the model while preserving the necessary information for the interaction detection. Experimental evaluation shows that, for practical interaction detection with three users, the proposed method achieves about 80% reduction in space and time, and is more scalable than the conventional ones especially for the increase of the number of users in the service.

key words: *feature interactions, telecommunication services, permutation symmetry, state reduction*

1. Introduction

Recent advancement of new telecommunication platform such as AIN (advanced intelligent network) enables functional enrichment in telecommunication services. As a result, various new services are being developed and deployed. When such new services (also called features, interchangeably) are added on the system, functional conflicts can happen between the new and existing services, which may trigger even system down. This conflict is generally known as *Feature Interaction*, and it becomes serious problem that prevents the rapid development of new services. *Feature interaction detection* is one of the most fundamental steps for interaction managements. It determines whether interactions occur or not for given pair of services.

The telecommunication services are often modeled by a finite state machine (i.e., state transition model), in which a global state consisting of user's local states successively moves to a next state by the occurrence of user's event. Then the interactions are defined on certain states in the FSM at which some undesirable properties hold.

The simplest way to detect the interaction in the state transition model is to exhaustively enumerate all possible states, then to identify such undesirable states that cause interaction. This approach is quite simple but powerful in the sense that all interactions are exactly detected. So, it is adopted by most of conventional detection frameworks [3]–[5], [8], [12]. However, due to concurrent characteristics of telecommunication services, the number of states in the model is exponential in the numbers of features and users, which is so-called *state explosion problem*. Therefore, the application of this approach should be limited to relatively simple services with small number of users. In order to make it possible to deal with more complex services, it is necessary to reduce the state space needed for the interaction detection in a certain manner.

As for the state reduction, two policies can be considered, which we call *efficiency-oriented* and *effectiveness-oriented*. First, the efficiency-oriented reduction mainly focuses on the large reduction of the state space rather than the optimal interaction detection. Cameron et al. [2] proposed the tool CADRES-FI which utilizes state abstraction based on heuristics. Kimbler introduced a concept of interaction filtering [9] that makes rough pre-evaluation of the interaction-prone service combination before the detection process. There also exist *static* interaction detection methods, in the sense that they never enumerate reachable states within the state transition model. Nakamura et al. [11] proposed a Petri Net based method using only necessary condition for the non-deterministic interactions. Also, a method by Yoneda et al. [14] reduces the complexity only by using the structure of a rule-based specification, STR [12], without reachable state enumeration.

In general, the efficiency-oriented reduction will attain drastic state reduction and semi-optimal detection as shown in [11]. Instead, the detection algorithms may miss some interactions, or may detect the redundant interactions which do not actually occur. Also, the types of detectable interactions may be limited. This may require examination of the detection results by subjective experts.

On the other hand, the effectiveness-oriented reduction attempts to reduce the state space with completely preserving necessary information for the interaction detection. To the best of our knowledge, the feature interaction detection based on this policy has not

Manuscript received January 4, 1999.
Manuscript revised May 14, 1999.

[†]The authors are with the Department of Informatics and Mathematical Science, Graduate School of Engineering Science, Osaka University, Toyonaka-shi, 560–8531 Japan.

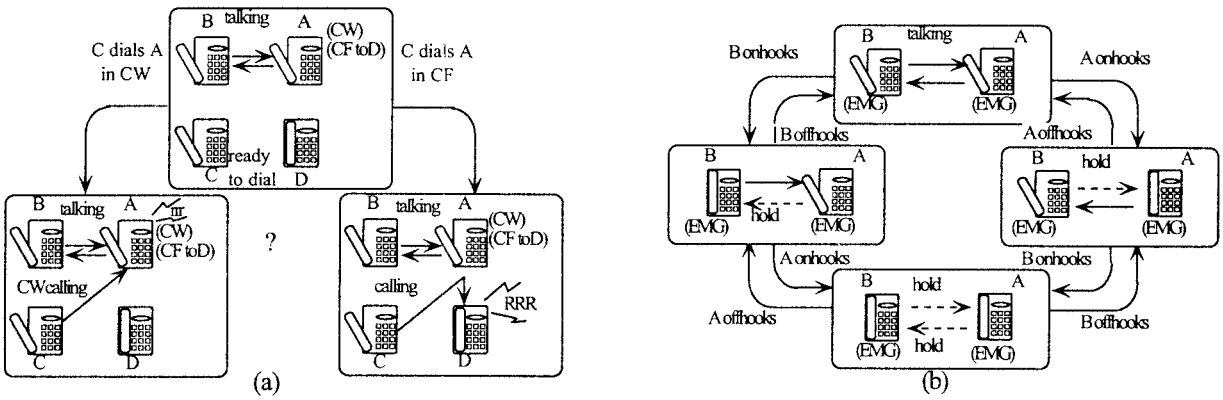


Fig. 1 Interaction examples.

been proposed yet, but there are several state reduction techniques based on this policy such as partial order reduction [6] and symbolic model checking [10] in other research fields. The effectiveness-oriented reduction will realize the *exact* (optimal) interaction detection and it can be applied to any types of interactions defined on the FSM. Instead, the reduction ratio may not be so significant as that of the efficiency-oriented reduction. The relationship between two policies is clearly trade-off, and thus they should be chosen for different purposes.

In this paper, we propose a new effectiveness-oriented reduction method for the interaction detection, which is a modification of the conventional reachable state enumeration. The key idea to achieve the reduction is to utilize a permutation symmetry [7] with respect to users. In the telecommunication systems, there exists a specific constraint that all subscribers of a service X are guaranteed to be able to use the same functionality of X. Under this constraint, suppose that both users A and B are subscribers of X. If we know A's possible behavior on X, then we can infer B's behavior on X from A's. Therefore, we can discard the state transitions for B's behaviors since they can be reproduced from A's. Based on this idea, we define a relation symmetrical on the states and transitions for the state reduction, and then provide theorems for the detection of three types of interactions: deadlock, loop, and non-determinism [8], [12]. Using the proposed method, we can exactly detect any of three types of interactions with much smaller state space. This fact implies it can be applicable to more complex services with many users.

2. Practical Examples

We present here two practical examples of feature interactions. More instances can be referred to [1], [3], [13], [15].

Example 1: Interaction between Call Waiting and Call Forwarding.

Call Waiting (CW): This service allows the subscriber to receive additional call while he is talking. Suppose that x subscribes to CW. Even when x is busy talking with y, x can receive an additional call from third party z.

Call Forwarding (CF): This service allows the subscriber to have his incoming calls forwarded to another user. Suppose that x subscribes to CF and that x specifies z to be a forwarding address. Then, any incoming call to x is automatically forwarded to z.

Interaction CW&CF: Let A, B, C and D be subscribers of the telephone network. Assume that A subscribes to both CW and CF. Suppose that (1) A is talking with B, (2) C is ready to dial, and (3) D is in the A's forwarding address and is idle. Then, if C dials A, should the call from C to A be received by A's CW feature, or should be forwarded to D by CF feature? This non-determinism will make A confused, thus should be avoided (See Fig. 1 (a)).

Example 2: Interaction of Emergency call with itself.

Emergency call (EMG): This service is usually deployed on police station and fire station. In case of the emergency accident, the call will be held even when the caller mistakenly onhooks. Suppose that x is a police station on which EMG is deployed, and that y made a call to x and is now busy talking with x. Then, even when y onhooks, the call is held instead of being disconnected. Followed by that, if y offhooks, the held line goes back to connected line and y can talk with x again. In order to disconnect the call, x has to onhook.

Interaction EMG_A & EMG_B: Suppose that both A and B subscribe to EMG and are talking to each other. Here, if A onhooks, the call is held by B's EMG. At this time, if A offhooks, the call reverts to the talking state. On the other hand, if B onhooks, the call is also held by A's EMG without being disconnected. Symmetrically, this is true when B onhooks first. Thus, neither A nor$B$ can disconnect the call. As a result, the call falls into a trap from which it never returns to the idle state (See Fig. 1 (b)).

3. Definitions

3.1 Service Specification

In this paper, we adopt a *rule-based service specification* for a service description method. Since the rule-based description is rich in the modularity and easy to understand, it is studied towards practical use, as shown in STR [12][†] and declarative transition rules [4].

3.1.1 Notation

First, we define the syntax notation of the specification.

Definition 1: A service specification S is defined as $S = \langle U, V, P, E, R, s_0 \rangle$, where

(a) U is a set of *constants* representing service users.
(b) V is a set of *variables*.
(c) P is a set of *predicate symbols*.
(d) E is a set of *event symbols*.
(e) R is a set of *rules*.
(f) s_0 is the (*initial*) *state*.

Each rule $r \in R$ is defined as follows:

r: pre-condition [event] post-condition.

Pre(post)-condition is a list of *predicates* $p(x_1, \ldots, x_k)$'s, where $p \in P$,$x_i \in V$ and k is called *arity* which is a fixed number for each p. Especially, pre-condition can include negations $\neg p(x_1, \ldots, x_k)$'s which implies $p(x_1, \ldots, x_k)$ does not hold. *Event* is a predicate $e(x_1, \ldots, x_k)$, where $e \in E$, $x_i \in V$. For convenience, we represent pre-condition, event and post-condition of rule r as $Pre[r]$, $Ev[r]$ and $Post[r]$, respectively.

A *state* is defined as a list of *instances of predicates* $p(a_1, \ldots, a_k)$'s, where $p \in P$,$a_i \in U$. We think of each state as representing a *truth valuation* [10] where instances in the list are true, and instances not in the list are false.

Example 3: Figure 2 shows an example of a service specification for Plain Ordinary Telephone Service (POTS). For example, $pots_3$ means that "Suppose that user x receives dialtone and y is not idle. At this time, if x dials y, then x will receive a busytone". State s_0 means that two users A and B are idle. In state s_0, two instances $idle(A)$ and $idle(B)$ are true because they are included in s_0. On the other hand, any other instances (e.g., $dialtone(B)$ or $calling(A, B)$) are false since they are not included in s_0.

3.1.2 State Transition Model

Next, we define the state transition model specified by the rule-based specification.

$$
\begin{array}{lcl}
U & = & \{A, B\} \\
V & = & \{x, y\} \\
P & = & \{idle, dialtone, calling, busytone, talk\} \\
E & = & \{offhook, onhook, dial\} \\
R & = & \{ \\
 & & pots_1 : idle(x)\ [offhook(x)]\ dialtone(x) \\
 & & pots_2 : dialtone(x)\ [onhook(x)]\ idle(x) \\
 & & pots_3 : dialtone(x), \neg idle(y)\ [dial(x, y)]\ busytone(x) \\
 & & pots_4 : dialtone(x), idle(y)\ [dial(x, y)]\ calling(x, y) \\
 & & pots_5 : calling(x, y)\ [onhook(x)]\ idle(x), idle(y) \\
 & & pots_6 : calling(x, y)\ [offhook(y)]\ talk(x, y), talk(y, x) \\
 & & pots_7 : talk(x, y), talk(y, x)\ [onhook(x)]\ idle(x), busytone(y) \\
 & & pots_8 : busytone(x)\ [onhook(x)]\ idle(x)\ \} \\
s_0 & = & idle(A), idle(B)
\end{array}
$$

Fig. 2 Rule-based specification for POTS.

Definition 2: Let $S = \langle U, V, P, E, R, s_0 \rangle$ be a service specification. For $r \in R$, let $x_1, \ldots, x_n$ ($x_i \in V$) be variables appearing in r, and let $\theta = \langle x_1 | a_1, \ldots, x_n | a_n \rangle$ ($a_i \in U$) be a substitution replacing each x_i in r with a_i. Then, an *instance* of r based on θ (denoted by $r\theta$) is defined as a rule obtained from r by applying $\theta = \langle x_1 | a_1, \ldots, x_n | a_n \rangle$ to r.

Definition 3: Let s be a state. We say rule r is *enabled* for θ at s, denoted by $en(s, r, \theta)$, iff all instances in $Pre[r\theta]$ take true value at s (i.e., all instances are included in s). When $en(s, r, \theta)$ holds, *next state* s' of s can be generated by deleting all instances in $Pre[r\theta]$ from s and adding all instances in $Post[r\theta]$ to s. For convenience, we describe it by

$$s' = s - Pre[r\theta] + Post[r\theta]$$

where $+$ and $-$ respectively represent addition and deletion operators on the list. These operators work in the same way as union and subtraction operators on the set, respectively. At this time, we say a *state transition* from s to s' caused by an event $Ev[r\theta]$ is defined on S, which is denoted by $s - Ev[r\theta] \rightarrow s'$ (or simply $s \rightarrow s'$). We say that state s is *reachable* from s_0 iff $s = s_0$ or a sequence of state transitions $s_0 - e_0 \rightarrow s_1$, $s_1 - e_1 \rightarrow s_2$, $\ldots$, $s_n - e_n \rightarrow s$ exists, which is denoted by $s_0 \rightarrow^* s$ (reflexive and transitive closure of $\rightarrow$).

Example 4: Let us consider rule $pots_1$ and state s_0 in Fig. 2. For a substitution $\theta = \langle x | A \rangle$, $en(s_0, pots_1, \theta)$ holds since $idle(A)$ in $Pre[pots_1\theta]$ is included in s_0. Then, next state s_1 can be defined as follows:

$$
\begin{array}{rcl}
s_1 & = & s_0 - Pre[pots_1\theta] + post[pots_1\theta] \\
 & = & idle(A), idle(B) - idle(A) + dialtone(A) \\
 & = & dialtone(A), idle(B)
\end{array}
$$

As a result, a state transition $s_0 - offhook(A) \rightarrow s_1$ is defined, which implies that if A offhooks at s_0, then A receives a dialtone, while B remains idle.

[†]Note that the notation in the following is similar to STR, but is slightly different from STR.

3.2 Feature Interactions

In this paper, we especially focus on the following three types of interactions. These are very typical cases of interactions and are discussed in many papers (e.g., [5], [8], [12]):

deadlock: Functional conflicts of two or more services cause a mutual prevention of their service execution, which results in a deadlock.

loop: The service execution is trapped into a loop from which the service execution never returns to the initial state.

non-determinism: An event can simultaneously activate two or more functionalities of different services. As a result, it cannot be determined exactly which functionality should be activated.

Before formally defining the above interactions, we define a combined operator of two service specifications. Because of the good modularity of the rule-based specification, we can easily combine two specifications.

Definition 4: For two specifications $S_1 = \langle U_1, V_1, P_1, E_1, R_1, s_{10}\rangle$ and $S_2 = \langle U_2, V_2, P_2, E_2, R_2, s_{20}\rangle$, we define a *combined specification* $S_1 \oplus S_2 = \langle U, V, P, E, R, s_0\rangle$ such that $U = U_1 \cup U_2$, $V = V_1 \cup V_2$, $P = P_1 \cup P_2$, $E = E_1 \cup E_2$, $R = R_1 \cup R_2$ and $s_0 = s_{10} + s_{20}$ where $+$ denotes the addition operator in Definition 3.

For each of three types of interactions mentioned above, we define the *undesirable states* each of which causes the corresponding interaction.

Definition 5: Let $S = \langle U, V, P, E, R, s_0\rangle$ be a given service specification. For any state s such that $s_0 \rightarrow^* s$, s is said to be a

deadlock state: iff $\neg en(s, r, \theta)$ holds for any $r \in R$ and θ.

loop state: iff there exists such a state s' that $s \rightarrow^* s' \rightarrow^* s$ and $\neg(s \rightarrow^* s_0)$.

non-deterministic state: iff there exists a pair of rules $r, r' \in R$ such that $en(s, r, \theta)$ and $en(s, r', \theta')$ hold, and that $Ev[r\theta] = Ev[r'\theta']$.

A service specification S is called *safe* iff S is free from the above states. For two service specifications S_1 and S_2, we say that S_1 *interacts with* S_2 iff both S_1 and S_2 are safe and $S_1 \oplus S_2$ is not safe.

Example 5: We explain the non-deterministic state using an example. Consider the following two rules cw_4 and cf_{10}, and a state s.

$$cw_4 : CW(x), talk(x, y), dialtone(z)[dial(z, x)] \\ CW(x), talk(x, y), CWcalling(z, x).$$
$$cf_{10} : CF(y, z), dialtone(x), idle(z)[dial(x, y)] \\ CF(y, z), calling(x, z).$$
$$s = CW(A), CF(A, D), talk(A, B), talk(B, A), dialtone(C), \\ idle(D)$$

The rules cw_4 and cf_{10} respectively describe the functionality of CW and CF shown in Example 1. Also, the state s means the situation of interaction. Let $\theta_1 = \langle x|A, y|B, z|C\rangle$ and $\theta_2 = \langle x|C, y|A, z|D\rangle$. Then, $en(s, cw_4, \theta_1)$ and $en(s, cf_{10}, \theta_2)$ hold, and $Ev[cw_4\theta_1] = Ev[cf_{10}\theta_2] = dial(C, A)$. This is just the interaction explained in Example 1.

4. Classical Approach

Most of the conventional feature interaction detection methods [3]–[5], [8], [12] adopt the exhaustive search, which is realized by means of a reachability graph.

4.1 Full Reachability Graph FRG

For a given service specification $S = \langle U, V, P, E, R, s_0\rangle$, the rule applications from initial state s_0 construct a finite state machine (FSM) consisting of all reachable states from s_0, in which a state moves to the next state by occurrence of an event. Since an FSM can be described by a labeled directed graph, here we directly define such an FSM as a directed graph. In the following, we represent a directed edge from s to s' labeled by e as a triple (s, e, s').

Definition 6: A labeled directed graph is defined as $G = \langle N, L, T\rangle$ where:

(a) N is a set of *nodes*.
(b) L is a set of *labels* attached to the directed edges.
(c) $T \subseteq N \times L \times N$ is a set of *directed edges*.

For any directed graph G, a *directed path* ρ is a sequence of directed edges: $\rho = (s_1, e_1, s_2), (s_2, e_2, s_3), \ldots, (s_n, e_n, s_{n+1})$. For this, the node s_1 is called *head node* of ρ, while s_{n+1} is called *tail node* of ρ, n is called a *length* of ρ. A directed path is a *directed cycle* iff its head node and tail node are identical. A node is called a *terminal* iff it has no outgoing edge.

Definition 7: Let $S = \langle U, V, P, E, R, s_0\rangle$ be a service specification. A *full reachability graph* for a given S is a labeled directed graph $FRG(S) = \langle N, L, T\rangle$ such that:

(a) $N = \{s | s_0 \rightarrow^* s\}$.
(b) L is a set of all instances of events.
(c) $T = \{(s, Ev[r\theta], s') |\ \ s - Ev[r\theta] \rightarrow s'\ \}$

Example 6: Figure 3 shows a full reachability graph for the POTS specification in Fig. 2. We can see there are 12 reachable states (represented by ovals) and 30 state transitions (represented by directed arrows).

4.2 Proof Rules of Interaction Detection

Using the full reachability graph FRG, we can easily identify the undesirable states in Definition 5.

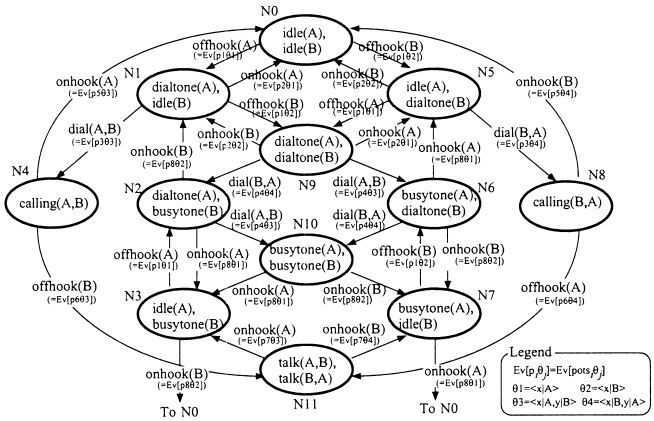


Fig. 3 FRG for POTS specification.

Proposition 1: The following properties are satisfied for $FRG(S)$.

(a) s is a terminal. $\Leftrightarrow$ s is a deadlock state.
(b) there exists a directed cycle starting from s, and there exists no directed path from s to s_0. $\Leftrightarrow$ s is a loop state.
(c) s has a pair of outgoing edges (s, e_1, s') and (s, e_2, s'') such that $e_1 = e_2$. $\Leftrightarrow$ s is a non-deterministic state.

Proof: Straightforward from Definitions 5 and 7. □

Thus, in order to detect the interactions between given two specifications S_1 and S_2, we first construct $FRG(S_1 \oplus S_2)$, then identify the undesirable states using Proposition 1.

Although the detection using FRG is quite simple and powerful, it may suffer from the *state explosion problem*. That is, the size of the FRG exponentially grows when the number of users and the number of rules in the specification become large. Therefore, the application of FRG is limited to relatively simple services with small number of users.

5. New Approach

In order to cope with the state explosion problem, we try to reduce the size of FRG without losing the necessary information. The key idea to achieve this reduction is to utilize a relation among states called permutation symmetry. The permutation symmetry was originally proposed in several fields such as colored Petri Net verification [7], but we show in this paper that the state reduction is successfully realized by means of a specific constraint on telecommunication services.

5.1 Key Idea

There exists the following specific constraint in the telecommunication services:
Constraint: All subscribers of service X must be able to use functionality of X in the same way.

For example, let us consider that A uses POTS's dialing functionality to B: "Suppose that A receives a dialtone and B is idle. At this time, if A dials B, then A will be calling B". Similarly, when B uses this functionality, the situation is: "Suppose that B receives a dialtone and A is idle. At this time, if B dials A, then B will be calling A." We can easily convince that two situations are symmetrical with respect to users, that is, the one can be inferred from the other only by swapping A and B. In terms of our service specification, we can observe the symmetry on states and state transitions.

Example 7: Consider the following states s_1 and s_2:

$$s_1 = dialtone(A), idle(B), busytone(C)$$
$$s_2 = dialtone(C), idle(A), busytone(B)$$

We see s_1 and s_2 are symmetrical, in the sense that s_2 is obtained from s_1 just by substituting A for C, B for A, C for B. In other words, letting $U = \{A, B, C\}$, there exists a permutation $\phi : U \to U$ such that $\phi(A) = C$, $\phi(B) = A$, $\phi(C) = B$ from s_1 to s_2. Now, let us apply the following rule to s_1 and s_2:

$$pots_3 : dialtone(x), idle(y)[dial(x, y)]calling(x, y).$$

Then, for $\theta_1 = \langle x|A, y|B\rangle$ and $\theta_2 = \langle x|C, y|A\rangle$, $en(s_1, pots_3, \theta_1)$ and $en(s_2, pots_3, \theta_2)$ hold.

$$pots_3\theta_1 : dialtone(A), idle(B)[dial(A, B)]calling(A, B).$$
$$pots_3\theta_2 : dialtone(C), idle(A)[dial(C, A)]calling(C, A).$$

As a result, the following next states s'_1 and s'_2 are obtained from s_1 and s_2, respectively.

$$s'_1 = calling(A, B), busytone(C)$$
$$s'_2 = calling(C, A), busytone(B).$$

Now, we can observe that there also exists the same permutation ϕ from s'_1 to s'_2. Roughly speaking, this fact implies that state transition $s_2 - dial(C, A) \to s'_2$ can be reproduced as $\phi(s_1) - \phi(dial(A, B)) \to \phi(s'_1)$ from $s_1 - dial(A, B) \to s'_1$. Therefore, we need no longer to store either states s_2, s'_2 nor transition $s_2 - dial(C, A) \to s'_2$ in the reachability graph.

From this example, we reach a hypothesis in general case that if we have a state transition $s - e \to s'$, then we also have $\phi(s) - \phi(e) \to \phi(s')$ for any permutation ϕ on U. If the hypothesis is true, then it is sufficient to have only $s - e \to s'$ in a reachability graph, since we can infer symmetric state transitions from $s - e \to s'$. As a result, the reduced reachability graph will preserve the all information of the original FRG.

5.2 Permutation Symmetry

In this section, we formally define the permutation symmetry. Throughout this section, we assume that a service specification $S = \langle U, V, P, E, R, s_0\rangle$ is given unless

especially specified. First, we define all permutations with respect to users.

Definition 8: Let $Perm(U)$ denote a set of all permutations $\phi : U \to U$. Each element ϕ of $Perm(U)$ is called a *permutation symmetry*.

Example 8: Let $U = \{A, B, C\}$. Then, $Perm(U) =$

$$\left\{ \begin{bmatrix} A & B & C \\ A & B & C \\ A & B & C \\ B & C & A \end{bmatrix} \begin{bmatrix} A & B & C \\ A & C & B \\ A & B & C \\ C & A & B \end{bmatrix} \begin{bmatrix} A & B & C \\ B & A & C \\ A & B & C \\ C & B & A \end{bmatrix} \right\}$$

Each permutation symmetry of $Perm(U)$ specifies a bijection ϕ from U to U. For example, $\begin{bmatrix} A & B & C \\ C & A & B \end{bmatrix}$ specifies a bijection ϕ such that $\phi(A) = C$, $\phi(B) = A$, $\phi(C) = B$.

Next, we extend $\phi \in Perm(U)$ for states, rules and substitutions.

Definition 9: Let $\phi \in Perm(U)$ be a permutation symmetry. Then, for any instance $p(a_1, \ldots, a_k)$ of a predicate with $p \in P, a_i \in U$, we define $\phi(p(a_1, \ldots, a_k)) = p(\phi(a_1), \ldots, \phi(a_k))$. At this time,

(a) For any state s, we define $\phi(s)$ to be a state obtained by applying ϕ to each instance of predicate in s.
(b) For any instances $r\theta$ of rule $r \in R$, we define $\phi(r\theta)$ to be an instance of rule obtained by applying ϕ to each instance of predicate in $r\theta$. Similarly, we define $\phi(Pre[r\theta])$, $\phi(Post[r\theta])$ and $\phi(Ev[r\theta])$.
(c) For any substitution $\theta = \langle x_1|a_1, \ldots, x_n|a_n\rangle$, $x_i \in V$, $a_i \in U$, we define $\phi(\theta) = \langle x_1|\phi(a_1), \ldots, x_n|\phi(a_n)\rangle$.

Two states s_1 and s_2 are *symmetrical*, denoted by $s_1 \approx s_2$ iff $\phi(s_1) = s_2$ for some $\phi \in Perm(U)^\dagger$.

Example 9: Consider again Example 7. Then, for a permutation symmetry $\phi = \begin{bmatrix} A & B & C \\ C & A & B \end{bmatrix}$, we can see that (a) $\phi(s_1) = s_2$ and $\phi(s_1') = s_2'$, (b) $\phi(pots_3\theta_1) = pots_3\theta_2$, and (c) $\phi(\theta_1) = \langle x|\phi(A), y|\phi(B)\rangle = \langle x|C, y|A\rangle = \theta_2$.

Before showing the main theorem, we present the following lemma. Intuitively, Lemma 1 implies that it does not matter whether we use a permutation symmetry ϕ before or after the instantiation of the rule r based on the substitution θ.

Lemma 1: Suppose that a permutation symmetry $\phi \in Perm(U)$ is given. Then, for any rule $r \in R$ and any substitution θ, $\phi(r\theta) = r\phi(\theta)$ holds. Also, $\phi(Pre[r\theta]) = Pre[r\phi(\theta)]$, $\phi(Post[r\theta]) = Post[r\phi(\theta)]$, $\phi(Ev[r\theta]) = Ev[r\phi(\theta)]$ hold.

Proof: See Appendix. □

Now, we are ready to provide the main theorem for a state transition.

Theorem 1: For any permutation symmetry $\phi \in Perm(U)$,

$$s{-}Ev[r\theta]{\to}s' \Leftrightarrow \phi(s){-}\phi(Ev[r\theta]){\to}\phi(s').$$

Proof: ($\Rightarrow$) Assume that the state transition $s{-}Ev[r\theta]{\to}s'$ is defined on S. Since $en(s, r, \theta)$ holds, all predicates of $Pre[r\theta]$ are included in s from Definition 3. From Definition 9, all predicates of $\phi(Pre[r\theta])$ are clearly included in $\phi(s)$. From Lemma 1, all predicates of $Pre[r\phi(\theta)]$ are included in $\phi(s)$. This implies $en(\phi(s), r, \phi(\theta))$ holds. Hence, from Definition 3 and Lemma 1, the next state s^* of $\phi(s)$ is obtained as follows:

$$\begin{aligned} s^* &= \phi(s) - Pre[r\phi(\theta)] + Post[r\phi(\theta)] \\ &= \phi(s) - \phi(Pre[r\theta]) + \phi(Post[r\theta]) \\ &= \phi(s - Pre[r\theta] + Post[r\theta]) = \phi(s') \end{aligned}$$

Therefore, a state transition $\phi(s){-}Ev[r\phi(\theta)]{\to}\phi(s')$ is defined on S. From Lemma 1, $\phi(s){-}Ev[r\phi(\theta)]{\to}\phi(s')$ $= \phi(s){-}\phi(Ev[r\theta]){\to}\phi(s')$, as required.
($\Leftarrow$) Assume that $s^* - e^* \to s'^* = \phi(s) - Ev[r\phi(\theta)] \to \phi(s')$ is defined on S. Since ϕ is a bijection, there exists an inverse $\phi^{-1} \in Perm(U)$. By applying ϕ^{-1}, we have $\phi^{-1}(s^*) - \phi^{-1}(e^*) \to \phi^{-1}(s'^*) = s{-}Ev[r\theta]{\to}s'$. Hence, we must show that if a state transition $s^* - e^* \to s'^*$ is defined on S, then $\phi^{-1}(s^*) - \phi^{-1}(e^*) \to \phi^{-1}(s'^*)$ is defined on S. However, it directly follows from $\Rightarrow$. □

Theorem 1 clearly explains the hypothesis discussed in Sect. 5.1 is true. That is, if we have a state transition $s{-}e{\to}s'$ on S, then we can confirm that all of its symmetric transitions $\phi(s){-}\phi(e){\to}\phi(s')$'s are defined on S. Theorem 1 is easily extended for the sequences of state transitions.

Corollary 1: For any permutation symmetry $\phi \in Perm(U)$, the following properties hold.

(a) $s_1 - e_1 \to s_2 - e_2 \to \ldots \to s_n \Leftrightarrow$
$\phi(s_1) - \phi(e_1) \to \phi(s_2) - \phi(e_2) \to \ldots \to \phi(s_n)$
(b) $s_0{\to}^* s \Leftrightarrow \phi(s_0){\to}^*\phi(s)$.

Proof: Property(a) follows by repeated use of Theorem 1. Property(b) is a direct consequence of Property(a). □

Thus, if we have only one transition sequence τ on S, then we can confirm the existence of all sequences symmetrical with τ by Corollary 1.

5.3 Consistent Symmetry

For a state transition defined on S, Theorem 1 and Corollary 1 guarantee the existence of its symmetrical transitions for any permutation symmetry $\phi \in$

†The relation $\approx$ is an equivalence relation on states since (i) $s \approx s$ (reflexive), (ii) $s_1 \approx s_2$ implies $s_2 \approx s_1$ (symmetric) and (iii) $s_1 \approx s_2$ and $s_2 \approx s_3$ imply $s_1 \approx s_3$ (transitive).

$Perm(U)$. However, they never guarantee the reachability of symmetric states from the initial state s_0 of S. That is, even if $s_0 \rightarrow^* s$, we cannot generally conclude $s_0 \rightarrow^* \phi(s)$ since Corollary 1 only says $\phi(s)$ is reachable from $\phi(s_0)$. In order to assure the reachability of symmetric states, we must put a restriction on ϕ so that $\phi(s_0) = s_0$. We define such a permutation symmetry ϕ as a consistent symmetry.

Definition 10: Let $S = \langle U, V, P, E, R, s_0 \rangle$ be a given service specification. A permutation symmetry ϕ is a *consistent symmetry* iff $\phi(s_0) = s_0$. Let $CS(U, s_0)$ denote a set of all consistent symmetries, i.e.,

$$CS(U, s_0) = \{\phi | \phi \in Perm(U) \wedge \phi(s_0) = s_0\}$$

Two states s and s' are *consistently symmetrical*, denoted by $s \approx_c s'$, iff $\phi(s) = s'$ for some $\phi \in CS(U, s_0)$. For a given state s, the *symmetric class* of s, denoted by $[s]$, is defined as $[s] = \{s' | s \approx_c s'\}$. For $[s]$, s is called a *representative* of $[s]$.

Note that all theories in the previous section are still valid for the consistent symmetry ϕ since $CS(U, s_0) \subseteq Perm(U)$, and that the relation $\approx_c$ is an equivalence relation on states as discussed in $\approx$. Additionally, the consistent symmetry provides the following theorem, which guarantees the reachability from the initial state s_0.

Theorem 2: Let $\phi \in CS(U, s_0)$ be any consistent symmetry. Then, $s_0 \rightarrow^* s \Leftrightarrow s_0 \rightarrow^* \phi(s)$.

Proof: It follows from Corollary 1(b) assuming $\phi(s_0) = s_0$. □

Example 10: Consider the following initial state s_0.

$$s_0 = CW(A), CW(B), idle(A), idle(B), idle(C)$$

s_0 means all A, B and C are idle and A and B are subscribers of CW. Assume that the following state s is reachable from s_0.

$$s = CW(A), CW(B), talk(A, B), talk(B, A), CWcalling(C, A)$$

in which A is receiving CW calling from C. Now, we apply a permutation symmetry ϕ such that $\phi(A) = C, \phi(B) = B, \phi(C) = A$ to s. Then,

$$\phi(s) = CW(C), CW(B), talk(C, B), talk(B, C), CWcalling(A, C)$$

C was not a CW subscriber at s_0. Therefore, it is inconsistent that $\phi(s)$, in which C is receiving CW calling, is reachable from s_0. This is justified by a fact that ϕ is not a consistent symmetry, i.e., $\phi(s_0) \neq s_0$.

5.4 Symmetric Reachability Graph SRG

Here, we define a reachability graph called symmetric reachability graph SRG.

Definition 11: Let $S = \langle U, V, P, E, R, s_0 \rangle$ be a service specification. A symmetric reachability graph for a given S is a labeled directed graph $SRG(S) = \langle N, L, T \rangle$ such that

(a) N is a partition of the set of all reachable states based on equivalence relation $\approx_c$ (Let it be $N = \{[s]\}$).
(b) L is a set of instances of events, and
(c) $T = \{([s], Ev[r\theta], [s']) | \ s - Ev[r\theta] \rightarrow s^* \wedge (s^* \approx_c s')\}$

In SRG, each node represents a symmetric class $[s]$ with a representative (state) s, and each arc outgoing from $[s]$ represents a state transition which occurs at the representative s. Also, from Theorem 2, each node $[s]$ in $SRG(S)$ implies that all states belonging to the class $[s]$ are reachable from the initial state s_0. The following algorithm constructs a SRG for a service specification S.

SRG construction algorithm.
Input: $S = \langle U, V, P, E, R, s_0 \rangle$
Output: $SRG(S) = \langle N, L, T \rangle$
Procedure:
$Waiting = N := \{[s_0]\}; \quad L = T := \emptyset;$
Repeat
  select $[s]$ from $Waiting$;
  for any $r \in R$ s.t. $en(s, r, \theta)$ for some θ {
    generate the next state s' by applying r to s;
    add $Ev[r\theta]$ to L ;
    if $[s^*] \notin N$ s.t. $s^* \approx_c s'$ then {
      add $[s']$ to N ; add $[s']$ to $Waiting$;
      add $([s], Ev[r\theta], [s'])$ to T ; }
    else add $([s], Ev[r\theta], [s^*])$ to T; }
  delete $[s]$ from $Waiting$;
Until $Waiting = \emptyset$

Example 11: Figure 4 shows a symmetric reachability graph SRG for POTS specification in Fig. 2. Each of nodes $N4$, $N5$, $N6$ and $N7$ represents a symmetric class

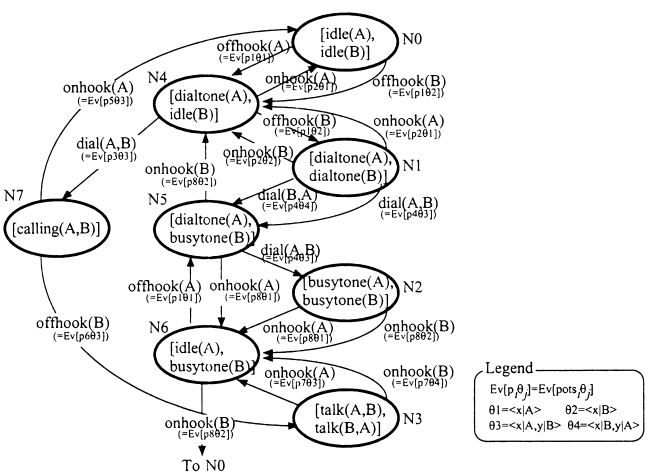


Fig. 4 SRG for POTS specification.

with two states, while each of other nodes represents a class with only one state. For example, $N4$ actually represents two symmetrical states $dialtone(A), idle(B)$ and $dialtone(B), idle(A)$, while $N0$ does one state $idle(A), idle(B)$. Compared with FRG in Fig. 3, SRG has smaller number of nodes (8 nodes) and edges (20 edges) (though it might be a small reduction in this example).

Proposition 2: The following properties are satisfied for $SRG(S)$.

(a) Each directed path ρ in $SRG(S)$: $\rho = ([s_1], e_1, [s_2]), ([s_2], e_2, [s_3]), \ldots, ([s_n], e_n, [s_{n+1}])$ has, for each state $s_1^* \in [s_1]$, a corresponding sequence of state transitions τ on S: $\tau = s_1^* - e_1^* \rightarrow s_2^* - e_2^* \rightarrow s_3^* - \ldots \rightarrow s_n^* - e_n^* \rightarrow s_{n+1}^*$, where $s_i \approx_c s_i^* (1 \leq i \leq n+1)$.

(b) Each sequence of transitions τ on S: $\tau = s_1 - e_1 \rightarrow s_2 - e_2 \rightarrow s_3 - \ldots \rightarrow s_n - e_n \rightarrow s_{n+1}$, where $s_0 \rightarrow^* s_1$, has a corresponding directed path in $SRG(S)$: $\rho = ([s_1^*], e_1^*, [s_2^*]), ([s_2^*], e_2^*, [s_3^*])), \ldots, ([s_n^*], e_n^*, [s_{n+1}^*])$, where $s_i \approx_c s_i^* (1 \leq i \leq n+1)$.

Proof: See Appendix. □

5.5 Proof Rules of Interaction Detection

Using the symmetric reachability graph SRG, we can identify the undesirable states in Definition 5.

Proposition 3: The following properties are satisfied for $SRG(S)$.

(a) $[s]$ is a terminal. $\Leftrightarrow$ each $s^* \in [s]$ is a deadlock state.

(b) there exists a directed cycle starting from $[s]$, and there exists no directed path from $[s]$ to $[s_0]$. $\Leftrightarrow$ each $s^* \in [s]$ is a loop state.

(c) $[s]$ has a pair of outgoing edges $([s], e_1, [s'])$ and $([s], e_2, [s''])$ such that $e_1 = e_2$. $\Leftrightarrow$ each $s^* \in [s]$ is a non-deterministic state.

Proof: Properties (a) (b) directly follow from Proposition 2 and Theorem 2.
Property (c) : ($\Rightarrow$) Assume that there exists a pair of the edges $([s], e, [s'])$ and $([s], e, [s''])$ in $SRG(S)$. From Definition 11(c), there exists a pair of transitions $s - e \rightarrow ss'$, $s - e \rightarrow ss''$ such that $s_0 \rightarrow^* s$, $ss' \in [s']$,$ss'' \in [s'']$. From Theorem 1, for any $\phi \in CS(U, s_0)$, there exists a pair of transitions $\phi(s) - \phi(e) \rightarrow \phi(ss')$, $\phi(s) - \phi(e) \rightarrow \phi(ss'')$. Since $s_0 \rightarrow^* s$, $s_0 \rightarrow^* \phi(s)$ from Theorem 2. Hence, from Definition 5, $s^* = \phi(s) \in [s]$ is a non-deterministic state, as required.
($\Leftarrow$) Since s^* is a non-deterministic state, there exists a pair of transitions $s^* - e \rightarrow ss'$, $s^* - e \rightarrow ss''$ such that $s_0 \rightarrow^* s^*$. From Definition 11 (a), there exists a node $[s]$ such that $s = \phi(s^*)$ in $SRG(S)$. From Theorem 1, there exists a pair of transitions $\phi(s^*) - \phi(e) \rightarrow \phi(ss')$, $\phi(s^*) - \phi(e) \rightarrow \phi(ss'')$ on S. From Definition 11(c), $SRG(S)$ has a pair of edges $([\phi(s^*)], \phi(e), [s'])$, $([\phi(s^*)], \phi(e), [s''])$, where $\phi(ss') \in [s']$,$\phi(ss'') \in [s'']$. Thus, $SRG(S)$ has a pair of edges $([s], e^*, [s'])$, $([s], e^*, [s''])$, as required. □

Thus, in order to detect the interactions between two specifications S_1 and S_2, we first construct $SRG(S_1 \oplus S_2)$, then identify the undesirable states using Proposition 3. The proof rules described in Proposition 3 is very similar to the conventional ones in Proposition 1, thus, they may appear to be not radically new concepts. However, the main contribution here is that we have shown that the similar detection mechanism can be still applicable in the reduced state space, due to well-organized theories concerning the SRG. This fact implies that the proposed method is easy to use and implement.

6. Experimental Evaluation

6.1 Preliminaries

For the experiments, we have developed a software to construct both SRG and FRG for any given rule-based specification. Also, based on [16], [17], we have prepared the rule-based specifications for the following six practical services (features): CW (Call Waiting), CF (Call Forwarding), DC (Direct Connect), DO (Denied Origination), DT (Denied Termination) and EMG (Emergency call). All of them commonly include POTS.

For each service X, we added a pair of rules which specify the registration and withdrawal of X. These rules enable all users to dynamically subscribe to and withdraw from X during the service execution. Also, we assume at the initial state s_0 that all users are idle and that none of users subscribes to any service yet, e.g.,

$$s_0 = yetCW(A), yetCW(B), yetCW(C), idle(A), idle(B), idle(C)$$

By doing this, we have $CS(U, s_0) = Perm(U)$. The experiments have been performed on the UNIX workstation Sun SS-UA1.

6.2 Experiment 1 (effectiveness and efficiency)

The objective of Experiment 1 is to evaluate the proposed method from the following two viewpoints:

(a) *effectiveness*: whether SRG can exactly identify all interactions detected by FRG,

(b) *efficiency*: how much reduction is attained by using SRG.

First, we checked if each of six specifications prepared in Sect. 6.1 is safe by constructing FRG (and SRG). As a result, we have found that all services except EMG

Table 1 Result of experiment 1.

Service	FRG(S)						SRG(S)					
Spec. S	DLK	LOP	NDT	|N|	|T|	Time(s)	DLK	LOP	NDT	|N|	|T|	Time(s)
CW+CF			x	102746	446124	4706.4			x	17610	76732	913.8
CW+DC				9592	38424	270.4				1684	6838	55.6
CW+DT			x	7120	39036	187.2			x	1344	7470	43.9
CW+DO				3480	16560	89.3				668	3234	20.7
CF+DC				65410	243876	1851.5				11065	41456	364.7
CF+DT			x	38584	206868	1006.1			x	6662	35902	213.2
CF+DO				17775	80538	447.6				3087	14079	93.5
DC+DT				5390	27510	69.6				954	4956	17
DC+DO			x	4654	23490	57.9			x	820	4202	14.2
DT+DO				1450	9180	15.7				300	1936	4.9
EMG		x		522	2802	3.5		x		116	646	1.3

are safe, while EMG contains the loop states as shown in Example 2. Next, we combined each pair of the remaining five services, then tried to detect the interactions between any two services by constructing FRG and SRG. For all specifications, the number of users is assumed to be three.

Table 1 summarizes the result. In the table, the columns DLK, LOP and NDT may contain "x" indicating the existence of deadlock states, loop states, nondeterministic states, respectively, $|N|$ and $|T|$ represent the number of nodes and edges in the graph, respectively. $Time(s)$ represents the execution time of the software for the construction of the graph.

It can be seen that all interactions detected by FRG are exactly identified by SRG, as is guaranteed by the theories. Also, we see that SRG attains about 80% reduction of FRG in both the graph size and the execution time (e.g., as for the number of nodes in $CW+CF$, $1-17610/102746=0.83$).

6.3 Experiment 2 (scalability)

In order to investigate the applicability of the proposed method to more complex services with many users, we compare SRG with FRG from the following viewpoints:

(a) *scalability w.r.t. # of users*: impact of the number of users on the graph size.
(b) *scalability w.r.t. # of features*: impact of the number of features on the graph size.

First we compare the scalability w.r.t. # of users. Concretely, for a fixed service specification, we observe the growth of the graph size by varying the number of users. We have selected the POTS specification described in Fig. 2 and varied the number of users from 2 to 8.

Figure 5 shows the result. Note the logarithmic scale on the vertical axis. It can be seen that, for the increase of the number of users, the size of FRG exponentially grows. We couldn't construct FRG with 8 users due to the memory overflow. On the other hand, the size of SRG grows much more slowly than that of FRG. The larger the number of users is, the more SRG attains the significant reduction. From this, we can say that our detection method by means of SRG is much more scalable than that of FRG with respect to the number of users.

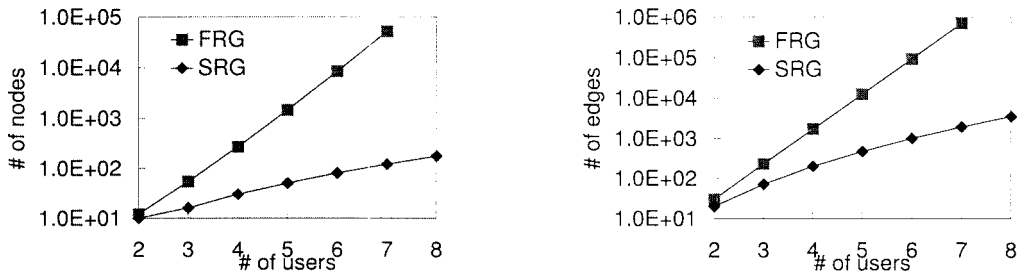
Next, we evaluate the scalability w.r.t. # of features. For this, we first prepare the POTS specification and fix the number of users (here we set it to be three). Then, by incrementally adding the specifications DO, DT, DC, EMG to POTS in this order, we observe how the graph size grows for each addition of the feature.

Figure 6 shows the result. Unfortunately, for the increase of the number of features, the size of SRG exponentially grows as well as the size of FRG. However, it can be seen that SRG constantly achieves more than 80% reduction of FRG (e.g., as for the number of nodes with 5 features, $1-58832/348868=0.83$). In this sense, SRG is slightly more scalable than FRG with respect to the number of features.

7. Discussion and Conclusion

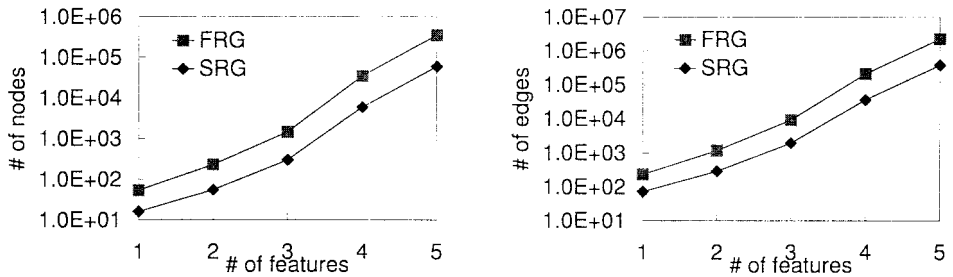
In this paper, we have newly proposed a state reduction technique using permutation symmetry for efficient feature interaction detection. The permutation symmetry works well under a constraint of telecommunication services that: all subscribers of service X must be able to use functionality of X in the same way. By using the proposed SRG, we can achieve not only efficient state space reduction but also exact interaction detection based on necessary and sufficient condition.

The result of Experiment 1 shows that the proposed method with SRG attains an adequate cost reduction for practical interaction detection. The scalability evaluated in Experiment 2 is the key factor for the state explosion problem. From the result, the proposed method is quite scalable even when the number of users increases. However, it has poor scalability with respect



(a) (b)

Fig. 5 Graph size w.r.t. # of users: (a) number of nodes (b) number of edges.



(a) (b)

Fig. 6 Graph size w.r.t. # of features: (a) number of nodes (b) number of edges.

to the number of features. This is not surprising since the permutation symmetry is defined only among users but not features, thus, it never contributes to state reduction when the number of users is fixed. Hence, we can conclude that state explosion caused by increase of the number of users is well circumvented by the proposed method. This facilitates interaction detection with many users, but with a relatively small number of feature combinations.

The state explosion problem concerning the number of features is still an open issue. One of the major sources of the state explosion is that we allow all users to dynamically subscribe to and withdraw from any feature in the experiment. This causes the combinational explosion of users' subscription cases to each feature. The promising way for this problem is to divide the whole state space based on the subscription condition, and then to separately explore the divided spaces in each of which the users' subscription condition is *statically fixed*. Also, symmetry among features will be powerful tool. For example, DT (Denied Termination) and TCS (Terminating Call Screening) [16], [17] are similar features. The difference is that DT blocks any terminating call, while, TCS screens users in its screening list. Therefore, appropriate symmetry may be defined among similar functionalities. We believe that these approaches improve the scalability regarding the number of features, and are now investigating them as our future work.

We have discussed only three types of interactions in this paper. However, *SRG* can be surely applied to the detection of any other types of interactions if the interactions can be defined on the finite state machine. The application to other interactions is also our future work.

References

[1] E.J. Cameron, N.D. Griffeth, Y-J. Lin, M.E. Nilson, W.K. Schnure, and H. Velthuijsen, "A feature interaction benchmark for IN and Beyond," Proc. of Second Workshop on Feature Interactions in Telecommunications Systems, pp.1–23, IOS Press 1994.

[2] E.J. Cameron, K. Cheng, F-J. Lin, H. Liu, and B. Pinheiro, "A formal AIN service creation, feature interactions analysis and management environment: An industrial application," Proc. of Fourth Workshop on Feature Interactions in Telecommunications Systems, pp.342–346, July 1997.

[3] C. Capellmann, P. Combes, J. Pettersson, B. Renard, and J.L. Ruiz, "Consistent interaction detection – A comprehensive approach integrated with service creation," Proc. of Fourth Workshop on Feature Interactions in Telecommunications Systems, pp.183–197, July 1997.

[4] A. Gammelgaard and E.J. Kristensen, "Interaction detection, a logical approach," Proc. of Second Workshop on Feature Interactions in Telecommunications Systems, pp.178–196, 1994.

[5] Y. Harada, Y. Hirakawa, T. Takenaka, and N. Terashima, "A conflict detection support method for telecommunication service descriptions," IEICE Trans. Commun., vol.E75-B, no.10, pp.986–997, Oct. 1992.

[6] G.J. Holzmann, "The model cheker SPIN," IEEE Trans. Software Eng., vol.23, no.5, pp.279–295, May 1997.

[7] K. Jensen, "Coloured Petri Nets," EATCS Monographs on Theoretical Computer Science, vol.1-2, Springer Verlag, 1992.
[8] A. Khoumsi, "Detection and resolution of interactions between services of telephone networks," Proc. of Fourth Workshop on Feature Interactions in Telecommunications Systems, pp.78–92, July 1997.
[9] K. Kimbler, "Addressing the interaction problem at the enterprise level," Proc. of Fourth Workshop on Feature Interactions in Telecommunications Systems, pp.13–22, July 1997.
[10] K.L. McMillan, Symbolic Model Checking, Kluwer Academic Publishers, 1993.
[11] M. Nakamura, Y. Kakuda, and T. Kikuno, "Petri-net based detection method for non-deterministic feature interactions and its experimental evaluation," Proc. of Fourth Workshop on Feature Interactions in Telecommunications Systems, pp.138–152, July 1997.
[12] T. Ohta and Y. Harada, "Classification, detection and resolution of service interactions in telecommunication services," Proc. of Second Workshop on Feature Interactions in Telecommunications Systems, pp.60–72, 1994.
[13] S. Tsang, E.H. Magill, and B. Kelly, "An investigation of the feature interaction problem in networked multimedia services," Proc. of Third Communication Networks Symposium, pp.58–61, 1996.
[14] T. Yoneda, and T. Ohta, "A formal approach for definition and detection of feature interactions," Proc. of Fifth Workshop on Feature Interactions and Software Systems, pp.165–171, IOS Press, 1998.
[15] P. Zave, "Feature interactions and formal specifications in telecommunications," IEEE Computer, vol.26, no.8, pp.20–30, 1993.
[16] ITU-T Recommendations Q.1200 Series., Intelligent Network Capability Set 1 (CS1), Sept. 1990.
[17] Bellcore, LSSGR Features Common to Residence and Business Customers I, II, III, Issue 2, July 1987.

Appendix

[Proof of Lemma 1]

Suppose that $\phi = \begin{bmatrix} a_1 & a_2 & \dots & a_{|U|} \\ b_1 & b_2 & \dots & b_{|U|} \end{bmatrix} \in Perm(U)$, and $a_i, b_i \in U$ be given.

Let $p = p_m(x_{m1}, x_{m2}, \dots, x_{mk})$ be any predicate in rule r. If we apply $\theta = \langle x_1|a_{i1}, \dots, x_n|a_{in}\rangle$ to r, then p in r is instantiated to an instance $p' = p_m(a_{im1}, a_{im2}, \dots, a_{imk})$ in $r\theta$. Next, we apply ϕ to $r\theta$. Then p' in $r\theta$ is transformed into the following instance p'' in $\phi(r\theta)$: $p'' = p_m(\phi(a_{im1}), \phi(a_{im2}), \dots, \phi(a_{imk})) = p_m(b_{im1}, b_{im2}, \dots, b_{imk})$

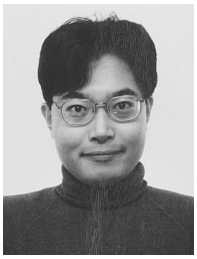
On the other hand, if we first apply ϕ to θ, we obtain $\phi(\theta) = \langle x_1|\phi(a_{i1}), \dots, x_n|\phi(a_{in})\rangle = \langle x_1|b_{i1}, \dots, x_n|b_{in}\rangle$ Next, by applying $\phi(\theta)$ to r, we get the following instance p^* of p in $r\phi(\theta)$: $p^* = p_m(b_{im1}, b_{im2}, \dots, b_{imk}) = p''$. Thus, for any predicate p in $r \in R$, two instances p'' in $\phi(r\theta)$ and p^* in $r\phi(\theta)$ are identical. Hence we conclude that $\phi(r\theta) = r\phi(\theta)$. By the similar discussions, we can prove that $\phi(Pre[r\theta]) = Pre[r\phi(\theta)]$, $\phi(Post[r\theta]) = Post[r\phi(\theta)]$ and $\phi(Ev[r\theta]) = Ev[r\phi(\theta)]$.

[Proof of Proposition 2]

We prove this lemma by means of induction over the length n of ρ/τ.

(Property(a):) When the length of ρ is zero, property(a) is clearly satisfied from Definition 11 (a). Next, assume that property(a) is satisfied when the length of ρ is n, and assume that we have a directed path ρ' such that $\rho' = ([s_1], e_1, [s_2]), \dots, ([s_n], e_n, [s_{n+1}]), ([s_{n+1}], e_{n+1}, [s_{n+2}])$. From the inductive hypothesis, it follows that, for each state $s_1^* \in [s_1]$, there exists a sequence of state transitions: $s_1^* - e_1^* \rightarrow s_2^* - \dots \rightarrow s_n^* - e_n^* \rightarrow s_{n+1}^*$. From Definition 11 (c) and the edge $([s_{n+1}], e_{n+1}, [s_{n+2}])$ of ρ', there exists a transition $s_{n+1} - e_{n+1} \rightarrow s'_{n+2}$, where $s'_{n+2} \in [s_{n+2}]$. From Theorem 1, for $\phi \in CS(U, s_0)$ such that $\phi(s_{n+1}) = s_{n+1}^*$, there exists $\phi(s_{n+1}) - \phi(e_{n+1}) \rightarrow \phi(s'_{n+2})$. Let $s_{n+2}^* = \phi(s'_{n+2})$ and $e_{n+1}^* = \phi(e'_{n+1})$. Since $s_{n+2} \approx_c s'_{n+2}$, $s_{n+2} \approx_c s_{n+2}^*$. Thus, there exists a transition $s_{n+1}^* - e_{n+1}^* \rightarrow s_{n+2}^*$. This means that $s_1^* - e_1^* \rightarrow s_2^* - \dots \rightarrow s_n^* - e_n^* \rightarrow s_{n+1}^* - e_{n+1}^* \rightarrow s_{n+2}^*$ is a sequence of transitions with the requested properties.

(Property(b):) When the length of τ is zero, property(b) is clearly satisfied from Definition 11 (a). Next, assume that property(b) is satisfied when the length of τ is n, and assume that we have a directed path τ' such that $\tau' = s_1 - e_1 \rightarrow s_2 - \dots \rightarrow s_n - e_n \rightarrow s_{n+1} - e_{n+1} \rightarrow s_{n+2}$, where $s_0 \rightarrow^* s_1$. From the inductive hypothesis, it follows that there exists a directed path: $([s_1^*], e_1^*, [s_2^*]), ([s_2^*], e_2^*, [s_3^*])), \dots, ([s_n^*], e_n^*, [s_{n+1}^*])$ and $s_{n+1} \approx_c s_{n+1}^*$. From a transition $s_{n+1} - e_{n+1} \rightarrow s_{n+2}$ of τ', we know $s_0 \rightarrow^* s_{n+2}$. From Definition 11 (a), there exists a node $[s_{n+2}^*]$ in $SRG(S)$ such that $s_{n+2}^* \approx_c s_{n+2}$. Moreover, from Theorem 1, for such that $\phi(s_{n+1}) = s_{n+1}^*$, there exists a transition $s_{n+1}^* - \phi(e_{n+1}) \rightarrow \phi(s_{n+2})$. Since $s_{n+2} \approx_c s_{n+2}^*$, $\phi(s_{n+2}) \approx_c s_{n+2}^*$. Hence, from Definition 11 (c), there exists an edge $([s_{n+1}^*], \phi(e_{n+1}), [s_{n+2}^*])$. This means that $([s_1^*], e1^*, [s_2^*]), \dots, ([s_n^*], e_n^*, [s_{n+1}^*]), ([s_{n+1}^*], e_{n+1}^*, [s_{n+2}^*])$ is a directed path in $SRG(S)$ with the required properties.



Masahide Nakamura received M.E. degree and Ph.D. degrees in computer science from Osaka University in 1996 and 1999, respectively. He is currently a Research Fellow of the Japan Society for the Promotion of Science (JSPS Research Fellow). He also received the Paper Award from the Telecommunication Advancement Foundation in 1997. His research interests include design, verification and testing of telecommunication services and communication protocols. He is a member of IEEE.

Tohru Kikuno was born in 1947. He received M.Sc. and Ph.D. degrees from Osaka University in 1972 and 1975, respectively. He was with Hiroshima University from 1975 to 1987. He is currently a Professor of the Department of Informatics and Mathematical Science at Osaka University. His research interests include analysis and design of fault-tolerant systems, quantitative evaluation of software development process and design of testing procedure of communication protocols. He has been active in the program committee of many international conferences such as FTCS, ISORC, ATS and RTSCA. He is a member of IPSJ, IEEE, ACM. He received the Paper Award from the Institute of Electronics, Information, and Communication Engineers of Japan in 1993.