Detecting Feature Interactions in Telecommunication Services by Bounded Model Checking

Tomoyuki Yokogawa
Osaka University
t-yokoga@ics.es.osaka-u.ac.jp

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

1. Introduction

Feature interaction refers to situations where a combination of different services behaves differently than expected from the single services' behaviors. In today's intelligent telecommunication networks, feature interaction is considered a major obstacle to the introduction of new features.

We propose a new interaction detection approach which uses *bounded model checking*. The central idea behind bounded model checking is to reduce the model checking problem to the propositional satisfiability (SAT) checking problem and to look for counterexamples that are shorter than some fixed length k for a given property.

In the literature, it has been reported that this method can work efficiently; but the original method does not work well for feature interaction detection, because a large formula is required to represent the transition relation, thus resulting in large execution time. We develop an alternative encoding method to address the problem.

2. Model

We adopt a variant of State Transition Rules (STR) to formally describe services. Each rule r is in the form $pre{-}condition\ [event]\ post{-}condition$. Figure 1 shows an example of a specification which describes the Plain Old Telephone Service (POTS).

$U = \{A, B\}$ // users
$V = \{x, y\}$ // variables
$P = \{idle(x), dialtone(x), busytone(x), calling(x, y), path(x, y)\}$
  // predicates
$E = \{onhook(x), offhook(x), dial(x, y)\}$ // events
$R = \{$
  $pots1 : idle(x)\ [offhook(x)]\ dialtone(x).$
  $pots2 : dialtone(x)\ [onhook(x)]\ idle(x).$
  $pots3 : dialtone(x), idle(y)\ [dial(x, y)]\ calling(x, y).$
  $pots4 : dialtone(x), \neg idle(y)\ [dial(x, y)]\ busytone(x).$
  $pots5 : calling(x, y)\ [onhook(x)]\ idle(x), idle(y).$
  $pots6 : calling(x, y)\ [offhook(y)]\ path(x, y), path(y, x).$
  $pots7 : path(x, y), path(y, x)\ [onhook(x)]\ idle(x), busytone(y).$
  $pots8 : busytone(x)\ [onhook(x)]\ idle(x).$
  $pots9 : dialtone(x)\ [dial(x, x)]\ busytone(x).$
  $\}$ // rules
$s_{init} = \{idle(A), idle(B)\}$ // initial state

Figure 1. Rule-based specification for POTS.

An STR specification specifies the state transition system defined as follows. An *instance* t of a rule is obtained by substituting users for variables of the rule. We represent the event and the post-condition of an instance t of a rule as $e[t]$ and $Post[t]$, respectively. In addition, we denote by $Pre[t]$ and $\hat{Pre}[t]$ the set of predicates in the pre-condition and the set of predicates whose negations are in the pre-conditions.

A *state* s is defined as a set of *instances of predicates*. We say that an instance of rule, t, is *enabled* for $e[t]$ at s iff all instances in $Pre[t]$ hold and no instances in $\hat{Pre}[t]$ hold at s. The execution of the enabled rule causes the *next state* $s' = (s \backslash Pre[t]) \cup Post[t]$. We define a relation $\xrightarrow{t}$ over states as follows: $s \xrightarrow{t} s'$ iff the execution of t causes s' from s.

3. Bounded Model Checking

To apply bounded model checking to service specifications, it is necessary to encode the state space and the transition relation by Boolean functions. Let $\mathcal{P} = \{p_1, \cdots, p_m\}$ and $\mathcal{T} = \{t_1, \cdots, t_n\}$ be the set of all instances of predicates and rules, respectively. A state s can then be viewed as a Boolean vector $s = (b_1, \cdots, b_m)$ such that $b_i = true$ iff an instance p_i of a predicate holds in that state.

Any set of states can be represented as a Boolean function such that $f(s) = true$ iff s is in the set. We say that f is the *characteristic function* of the state set.

The characteristic function $E_t(s)$ of the set of states where $t \in \mathcal{T}$ is enabled is

$$E_t(s) = \bigwedge_{p_i \in Pre[t]} b_i \wedge \bigwedge_{p_i \in \hat{P}re[t]} \neg b_i.$$

Any relation over states can be similarly encoded since they are simply sets of tuples. The relation $\xrightarrow{t}$ is represented as Boolean function $T_t(s, s')$ as follows.

$$E_t(s) \wedge \bigwedge_{p_i \in Post[t] \setminus Pre[t]} b'_i \wedge \bigwedge_{p_i \in Pre[t] \setminus Post[t]} \neg b'_i$$
$$\wedge \bigwedge_{p_i \in (\mathcal{P} \setminus (Pre[t] \cup Post[t])) \cup (Pre[t] \cap Post[t])} (b_i \leftrightarrow b'_i)$$

where $s' = (b'_1, \cdots, b'_m)$.

The traditional encoding uses a disjunction of $T_t(S, S')$ to represent one state transition. The formula would be very large in size, which becomes a major obstacle to applying bounded model checking to feature interaction detection.

Our scheme alleviates the above problem with a new encoding. Let $d_t(s, s') = T_t(s, s') \vee \bigwedge_{p_i \in \mathcal{P}} (b_i \leftrightarrow b'_i)$. Then $d_t(s, s')$ is

$$((\bigwedge_{p_i \in Pre[t]} b_i \wedge \bigwedge_{p_i \in \hat{P}re[t]} \neg b_i$$
$$\wedge \bigwedge_{p_i \in Post[t] \setminus Pre[t]} b'_i \wedge \bigwedge_{p_i \in Pre[t] \setminus Post[t]} \neg b'_i)$$
$$\vee \bigwedge_{p_i \in (Pre[t] \cup Post[t]) \setminus (Pre[t] \cap Post[t])} (b_i \leftrightarrow b'_i))$$
$$\wedge \bigwedge_{p_i \in (\mathcal{P} \setminus (Pre[t] \cup Post[t])) \cup (Pre[t] \cap Post[t])} (b_i \leftrightarrow b'_i)$$

It is easy to see that $d_t(S, S') = true$ iff $S \xrightarrow{t} S'$ or $S = S'$. $d_t(S, S')$ differs from $T_t(S, S')$ only in that $d_t(S, S')$ evaluates to true also when $S = S'$. Using this property, a transition (or more) can be represented by a conjunction of d_t. Let

$$\varphi = I(s_0)$$
$$\wedge d_{t_1}(s_0, s_1) \wedge d_{t_2}(s_1, s_2) \wedge \cdots \wedge d_{t_n}(s_{n-1}, s_n)$$
$$\wedge d_{t_1}(s_n, s_{n+1}) \wedge d_{t_2}(s_{n+1}, s_{n+2}) \wedge \cdots \wedge d_{t_n}(s_{2n-1}, s_{2n})$$
$$\cdots$$
$$\wedge d_{t_1}(s_{(k-1)*n}, s_{(k-1)*n+1}) \wedge \cdots \wedge d_{t_n}(s_{k*n-1}, s_{k*n})$$
$$\wedge f_G(s_{k*n})$$

where G is the set of states whose reachability is to be decided, and $f_G(S)$ and $I(S)$ are the characteristic functions for G and the initial state. If the formula φ is satisfiable, then we can conclude that there is a state in G that can be reached in at most $k * n$ steps, because φ evaluates to true iff (i) s_0 is the initial state, (ii) for any $0 \leq i < k * n$, $s_i \xrightarrow{t} s_{i+1}$ for some $t \in \mathcal{T}$ or $s_i = s_{i+1}$, and (iii) $s_{k*n} \in G$. Therefore s_{k*n}, which belongs to G, is reachable from the initial state in at most $k * n$ steps. An important observation here is that the method may be able to find a state in G that requires more than k transition executions to reach.

More importantly, φ can be converted into a much succinct formula that is not logically equivalent but has the same satisfiability. Let $s_i = (b_{1,i}, b_{2,i}, \cdots, b_{m,i})$. For each $d_t(s_j, s_{j+1})$ in φ, term $(b_{i,j} \leftrightarrow b_{i,j+1})$ can be omitted for all $p_i \in (\mathcal{P} \setminus (Pre[t] \cup Post[t])) \cup (Pre[t] \cap Post[t])$ by quantifying $b_{i,j+1}$ away. That is, d_t in φ can be replaced with

$$(\bigwedge_{p_i \in Pre[t]} b_i \wedge \bigwedge_{p_i \in \hat{P}re[t]} \neg b_i$$
$$\wedge \bigwedge_{p_i \in Post[t] \setminus Pre[t]} b'_i \wedge \bigwedge_{p_i \in Pre[t] \setminus Post[t]} \neg b'_i)$$
$$\vee \bigwedge_{p_i \in (Pre[t] \cup Post[t]) \setminus (Pre[t] \cap Post[t])} (b_i \leftrightarrow b'_i).$$

For example, when t is the instance of the rule $pots4$ in Figure 1 with substitution $(x, y) = (A, B)$, the above formula will be $(dialtone(A) \wedge \neg idle(B) \wedge busytone(A)' \wedge \neg dialtone(A)') \vee ((dialtone(A) \leftrightarrow dialtone(A)') \wedge (busytone(A) \leftrightarrow busytone(A)'))$.

Table 1. Performance of bounded model checking.

	k	Time(sec)	Trad. scheme	length
CW+CF	2	3.02	4934.76	10
CW+DT	3	4.81	212.10	8
CW+OCS	2	2.90	330.15	8
CW+TCS	2	3.80	1470.37	8
CF+DT	2	0.02	53.52	5
CF+OCS	2	0.02	89.32	5
CF+TCS	2	0.02	65.10	5
DC+DO	1	0.02	0.87	2
DT+OCS	2	0.05	1.91	3
DT+TCS	1	0.02	1.86	3
OCS+TCS	1	0.01	1.01	2

CW:Call Waiting, CF:Call Forwarding, DC:Direct Connect, DO:Denied Origination, DT:Denied Termination, OCS:Originating Call Screening, TCS:Terminating Call Screening.

4. Experimental Results

We conducted experimental evaluation for the seven services. Combining two of the seven services, we examined a total of the 21 pairs.

The experiments were performed on a Linux workstation with a 853 MHz Pentium III processor. The number of users was assumed to be four. ZChaff was used as a SAT solver.

Of the 21 combinations, 11 can cause nondeterminism, a situation where an event can simultaneously activate two or more functionalities of different services. Table 1 compares the proposed encoding and the traditional one with respect to the running time to detect nondeterminism. Items in the 'length' column represent the length of the shortest path to the states that cause nondeterminism.

As can be seen in this table, when using the proposed encoding, interaction was detected with k of less than or equal to three for all cases. Note that the length of the shortest counterexample coincides with the smallest k value at which the traditional scheme can find such a computation. This resulted in large detection time of the traditional scheme, as shown in this table.