Formal Approaches for Detecting Feature Interactions, Their Experimental Results, and Application to VoIP

T. Yoneda, S. Kawauchii, J. Yoshida and T. Ohta

SOKA University
Contents

1. Background and Problems
2. Problems and Their Solutions
   - Terminal Assignments
   - Reachability Test
   - Static Detection Algorithm
3. Implementation and Evaluations
4. Application to VoIP
Background

• Dynamic Detection: Detecting interactions by executing service specifications.
  
  Explosion in computation time for detecting feature interactions

• Static Detection: Detecting interactions solely by analyzing service specifications
  
  Coverage and Redundancy in detecting feature interactions
Problems in Static Detection

Static Detection System

Generation of Testing Objects

Input Spec.

Comb. of Spec. → Terminal Assignments → Reachability Test → Detection of Interactions

Det. Fls

Prob. 1

Prob. 2

Prob. 3

Filtering

Formal Detection
Contents

1. Background and Problems
2. Problems and Their Solutions
   - Terminal Assignments
   - Reachability Test
   - Static Detection Algorithm
3. Implementation and Evaluations
4. Application to VoIP
Specification Description Language

STR: State Transition Rule

Specification is represented as a set of STR rules

Form: Pre-cond., Event: Event, Post-cond.

Pre-cond. and Post-cond. are represented as a set of primitives.

Rule application: Precondition exists in the current system state

State change:
Terminal Assignments

Terminal is described as a **variable** in rules.

To detect FIs, **real terminals are assigned** to variables.

**Ex.**:

- $m$-cfv($y_a, z_a$), dialtone($x_a$), idle($z_a$)
- $m$-cfv($B, C$), dialtone($A$), idle($C$)
- $m$-ocs($x_b, y_b$), dialtone($x_b$), idle($y_b$)
- $m$-ocs($A, C$), dialtone($A$), idle($C$)
Problems so far

It was not clear how to assign real terminals to terminal variables.

Explosive computation time with all assignments
Low coverage with reduced assignments

Proposal of terminal assignment method, where unnecessary terminal assignments are deleted.
Basic Idea for Terminal Assignment

No terminals belong to both services: no feature interactions
If a terminal belongs to both services, feature interactions may occur.

A terminal belongs to both services:
   xa for service A, xb for service B   xa=xb=terminal P

Combination of variables: a set of pairs of variables to which
   the same real terminals are assigned

Different terminal assignments to the same combination of variable
gives equivalent states, the same state with different terminal names.
Equivalent States

Case 1

Combination of variables are the same. Terminal assignments are different.
Basic Idea for Terminal Assignment

No terminals belong to both services: no feature interactions
If a terminal belongs to both services, feature interactions may occur.

A terminal belongs to both services:
\[ x_a \text{ for service A, } x_b \text{ for service B} \quad x_a = x_b = \text{terminal } P \]

Combination of variables: a set of pairs of variables to which the same real terminals are assigned

Different terminal assignments to the same combination of variable gives equivalent states, the same state with different terminal names.

Interactions caused in equivalent states are equivalent interactions.

One terminal assignment to one combination of variables. Consider only different combination of variables.
The number of combinations of variables

\[
\begin{align*}
\text{Real terminals} & \quad \begin{array}{c}
\text{A} \quad \text{B} \quad \text{C} \\
\text{Var. 1} \quad \text{Var. 2} \quad \text{Var. 3}
\end{array} & \quad \begin{array}{c}
\text{D} \quad \text{E} \\
\vdots
\end{array} \\
\text{Service A} & \quad \begin{array}{c}
\text{Var. 1} \quad \text{Var. 2} \quad \text{Var. 3} \\
\text{Var. 1} \quad \text{Var. 2}
\end{array} & \quad \text{:} \quad n_a \\
(n_a \geq n_b) & \quad n_b \\
\text{Service B} & \quad \begin{array}{c}
\text{Var. 3} \quad \text{Var. 4}
\end{array}
\end{align*}
\]

\[
N = \sum_{t=1}^{t=n_a} n_a C_t \times n_b P_t \quad \text{When } n_a = n_b = 3, \quad N = 33
\]

The number of all terminal assignments: 14400

\[
n_T P n_a \times n_T P n_b \quad \text{Here } n_T = n_a + n_b
\]
The number of terminal assignments for a service pair which have 3 term. variables.

<table>
<thead>
<tr>
<th>Num of real ter.</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>...</th>
<th>100</th>
</tr>
</thead>
<tbody>
<tr>
<td>(a)</td>
<td>14,400</td>
<td>44,100</td>
<td>112,896</td>
<td>254,016</td>
<td>518,400</td>
<td>980,100</td>
<td>1,742,400</td>
<td>...</td>
<td>941,288,040,000</td>
</tr>
<tr>
<td>(b)</td>
<td>33</td>
<td>33</td>
<td>33</td>
<td>33</td>
<td>33</td>
<td>33</td>
<td>33</td>
<td>...</td>
<td>33</td>
</tr>
</tbody>
</table>

(a) Before deletion
(b) After deletion
Problem 2

Reachability Test
New Method

Conventional

Generating states
P-invariant in Petri-Net

Require much time

New Proposal

Using knowledge which can be obtained easily
Illegal Combinations of Primitives

Ex. \{dialtone(x), idle(x)\}

Ex. \{dialtone(x), cw-calling(x,y)\}

considering only service i and POTS

Generating Knowledge for reachability test
for combined service of service a and service b

Knowledge

\{idle,dialtone\},\{dialtone,talk\},…
Problem 3

Static Detection Algorithm of Feature Interactions
Classification of Interactions

Classification based on FSM Model

**Logical Int.** can be identified by State Transition Diagrams
- Non-determinacy, Dead lock, Live lock

**Semantic Int.** can be identified by meaning of State Transition
- Occurrence of abnormal state/transition
- Disappearance of normal state/transition
**Static Detection Algorithm**

<table>
<thead>
<tr>
<th>Pre-cond</th>
<th>Event</th>
<th>Post-cond</th>
</tr>
</thead>
<tbody>
<tr>
<td>rule for service ( a )</td>
<td>( e_a : r_{an} )</td>
<td>rule for service ( b )</td>
</tr>
</tbody>
</table>

Current State

State for service \( a \) \( r_{ac} \)

State for service \( b \) \( r_b \)

Next State

State for service \( a \) \( r_{an} \)

Judging Formula:

\[
\{(e_a \neq e_b) \land [(r_{bc} - r_{ac}) \cup r_{an}] \neq (r_{bc} \cup \Delta r_{ac})] \\
\lor \{(e_a = e_b) \land [(r_{bc} - r_{ac}) \cup r_{an}] \neq (r_{bn} \cup \Delta r_{ac})] \\
\lor \{(r_{ac} - r_{an}) \notin (r_{bc} - r_{bn})]\}
\]

→ can be judged solely by specifications
Contents

1. Background and Problems
2. Problems and Their Solutions
   - Terminal Assignments
   - Reachability Test
   - Static Detection Algorithm
3. Implementation and Evaluations
4. Application to VoIP
Evaluation Items

• Coverage: As close as possible to 100 %

• Redundancy:
  Detecting what is not actually interaction

• Detection time:
Bench Mark

FIW98 contest results published in 2000

12 services: CFBL, CND, INFB, INFR, INTL, TCS, TWC, INCF, CW, INCC, RC, CELL

FIW2000 contest results could not used because of lack in detailed information: scenario where interactions occur.
Detection System

Knowledge

Specifications

Paring specifications

Terminal assignments

Reachability tests

Interaction detection

Interactions
Coverage and Redundancy

The number of interactions detected: 2,650

Including all interactions described in the benchmark

No redundancies: miss detection, duplicated detection
The number of testing subjects

reduced to 0.4 % by deleting equiv. term. assignments and reduced to 0.07 % after reachability test.
Detecting Time

- Mean time for one pair of services: 17.7 sec.
- Total time for 12 services: 23 min.
Evaluations

- Coverage: 100% based on the benchmark
- Redundancy: no redundancies
- Detection time: 17.7 sec.; mean time for a pair of services

Effective detection system
Contents

1. Background and Problems
2. Problems and Their Solutions
   - Terminal Assignments
   - Reachability Test
3. Implementation and Evaluations
4. Application to VoIP
Active Network for VoIP

Validation server

Upload to User terminal

GW

Active GK

GW

Active GK

IP Tel.

PSTN

PC

IP Tel.

PSTN

User terminal

Upload to Validation server

deliver

Upload to User terminal
Experimental System Structure for Validation Server

User terminal

Validation Server

FI Detection Part

Delivering user programs

Receiving user programs

Web server

Active GK

Active GK

Uploading programs
Future Work

• Interaction resolution algorithm
  – for selecting interactions
to be resolved actually
  – for automatic resolution or assisting resolution
• Application to other than telephone services
  – Home network
  – Ado-hoc network
  – Data base system
  – ...

Thank you for your kind attentions.
ESTR(2)

Syntax:

<table>
<thead>
<tr>
<th>Pre-condition</th>
<th>event</th>
<th>Post-condition</th>
<th>Action</th>
</tr>
</thead>
</table>

Pre-condition: conditions for state transition

Event: trigger for state transition

Post-condition: state after transition

Action: procedure accompanied by state transition

(send a signal, retrieve database, and so on)

Example;

idle(x) setup(x,y): w-alert(y,x), {Send(setup,x,y)}
Example for ESTR Description

idle(x) arq(x): w-setup(x),{Send(acf,x)}
w-setup(x) setup(x,y): w-arq(y,x),{Send(setup,x,y)}
w-arq(y,x) arq(y,x): w-proc(y,x),{Send(acf,y)}
w-proc(y,x) proc(y,x): w-alert(y,x),{Send(proc,y,x)}
w-alert(y,x) alert(y,x): w-conn(y,x),{Send(alert,y,x)}
w-conn(y,x) conn(y,x): talk(x,y),{Send(conn,y,x)}
talk(x,y) disc(x,y): w-release(y,x),{Send(disc,x,y)}
talk(x,y) disc(y,x): w-release(x,y),{Send(disc,y,x)}
w-release(y,x) release(y,x):
    w-release_conf(x), w-drq(y),{Send(release,y,x)}
w-release_conf(x) release_conf(x): w-drq(x),{}}
Structure of Active GK

Active GK

EE

Rule selection

Input of signal

Actions execution

Network platform provided by a vendor

Rule database

User Programs
Example for Interaction

Interaction between CFV and OCS

<CURRENT STATE><EVENT> <NEXT STATE>

Forward to C

CFV Service: Reject C
OCS Service: dialtone

dial(A,B)

Calling

Interaction
Comparison with Nakamura’s Method

Detection Time

- Can be reduced to 1 60th

**DT**: reject all terminating call
**DO**: reject all originating call
**DC**: direct call (hot line)
Deleting Equivalent Terminal Assignments

Terminal assignments after deleting equivalent ones:
One set of terminal assignment to a combination of terminal variables to which the same terminals are assigned.

The number of all terminal assignments:
\[ gP_{k_a} \times gP_{k_b} \]
Deleting equivalent terminal assignments:
\[ \sum_{t=0}^{k_a} P_t \times k_b \cdot C_t \quad (k_a \leq k_b) \]

\( g \): The number of real terminals to be assigned
\( k_a, k_b \): The number of terminal variables in service a and b, respectively
\( t \): The number of pairs of terminal variables to be assigned the same term
Static Detection Algorithm

Pre-cond Event : Post-cond

rule for service \( a \)

\( e_a : r_{ac} \)

\( r_{ac} \)

\( r_{an} \)

rule for service \( b \)

\( e : r_{b} \)

\( r_{bc} \)

Judging Formula:

\[
\{(e_a \neq e_b) \land \{(r_{ac} \cup r_{bc}) - r_{ac}\} \cup r_{an} \neq r_{bc} \cup \Delta r_{ac}\}\}
\]

\[
\lor \{(e_a = e_b) \land \{(r_{ac} \cup r_{bc}) - r_{ac}\} \cup \{r_{an} \neq (r_{bn} \cup \Delta r_{ac})\}\}
\]

\[
\lor \{(r_{ac} - r_{an}) \cup (r_{an} \neq r_{bc} - r_{bn})\}\]

→can be judged solely by specifications