An automatic off-line feature interaction detection method
by static analysis of specifications

Hélène Jouve, Pascale Le Gall and Sophie Coudert

ICFI’05 Leicester

France Telecom R&D
Université d’Évry - LaMI
ENST - Laboratoire SoC
Service: extension of a basic system

Example: POTS, CFB, TCS, CW, etc.

Separately designed but integrated on the same network => interactions

Solve or prevent interactions => detect them before integration of services of at runtime

New challenges: internet telephony
1. **Detect** interactions,

2. **At the earliest**, from **graphical specifications**
   - minimize impact of conflicts
   - graphical is easily readable

3. **Automatically**
   - without supplementary knowledge than the specifications

4. **Give an explanation** of detected conflict
   - Helps solving conflicts or modify specification
1. Our method for phone services

2. Current and future works: SIP
Our choices

- operational and graphical specifications
  - compromise between expressiveness and feasibility of computations
  - user point of view, abstraction of the network
- simple properties
  - formulas such as $emitting(A, ringingtone) \Rightarrow \neg idle(A)$
- No integration, combination choices, no computed model
  - static method
Example of specification: focus on CFB

VAR A: POTS
VAR B: CFB(C)
VAR C: POTS

start(rgtone).B
start(rbtone).A
A.call(B)

POTS, 10, (A)
POTS, 5, (A, C)
POTS, 5, (A, B)
dialing(A)
idle(B)
idle(C)

A.onhook

start(lbtone).A

idle(B)
idle(C)

POTS, 5, (A, B)
POTS, 5, (A, C)
POTS, 10, (A)

idle(A)

• Aliases
• Subscriptions and variables declarations
• Determinism of transitions
• Invariant properties
Main steps of our method:

1. Computation of trigger information of \( f \)
2. Trigger information of \( f' \)
3. Computation of interactions
4. Interactions between \( f \) et \( f' \)
Computing and modelling trigger information

- Step 1: suppress aliases (except loops)
- Step 2: pair branches and extract non paired branches (trigger)
- Step 3: annotate non paired branches with "intentions"
Computing trigger information: pairing diagrams

VAR A:POTS

VAR B:POTS

VAR A:POTS

VAR B:CFB(C)

VAR C:POTS

emitting(A, rbtone)

emitting(B, rgtone)

emitting(A, lbtone)

dialing(A)

start(rbtone).A

start(rgtone).B

start(lbtone).A

A.call(B)

A.onhook

idle(B)

idle(A)

VAR A:POTS

VAR B:POTS

VAR A:POTS

VAR B:CFB(C)

VAR C:POTS

emitting(A, rbtone)

emitting(B, rgtone)

emitting(A, lbtone)

dialing(A)

start(rbtone).A

start(rgtone).B

start(lbtone).A

A.call(B)

A.call(B)

A.onhook

A.onhook

idle(B)

idle(A)

idle(B)

idle(A)

idle(B)

idle(C)

idle(A)

idle(B)

idle(C)

idle(A)

start(lbtone).A

start(rgtone).C

start(lbtone).A

start(rgtone).B

start(rgtone).C
Computing trigger information: intentions
Computing interactions: direct and indirect at trigger

1. direct interactions:
   If $\text{Compat}^\sigma(C, C')$
   
<table>
<thead>
<tr>
<th>Subs.</th>
<th>$\sigma(A) \cup A'$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cond.</td>
<td>$\sigma(C) \cup C'$</td>
</tr>
<tr>
<td>Msg.</td>
<td>$\sigma(m) = m'$</td>
</tr>
<tr>
<td>Conflict</td>
<td>$\sigma(S)$ ou $S'$</td>
</tr>
</tbody>
</table>

2. indirect interactions
   using intentional message
Computing interactions: generalization

Path

\[ C \]

\[ m \]

\[ A \]

\[ S \]

\[ m_i \]

\[ ci \]

\[ C_f \]

\[ m_f \]

\[ A_f \]

\[ S_f \]

\[ D \]

Trigger information of \( f \)

\[ C' \]

\[ m' \]

\[ A' \]

\[ S' \]

\[ D' \]

Trigger information of \( f' \)
Example of computed interactions (RBF vs TM and RBF vs CW)

<table>
<thead>
<tr>
<th>S</th>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>dialing(A)</td>
<td>idle(B)</td>
<td></td>
</tr>
<tr>
<td>m</td>
<td>A.call(B)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>startplaying(RBFmsg).A</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>startplaying(TMaccueilmsg).A</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>S</th>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>dialing(A)</td>
<td>talking(B,C)</td>
<td></td>
</tr>
<tr>
<td>m</td>
<td>A.call(B)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>startplaying(RBFmsg).A</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>start(CWtone).B</td>
</tr>
</tbody>
</table>
RBF vs CW

- \{dialing(A), \sim idle(B)\} \text{, and}\n  \{talking(B, C')\}\text{ are compatible}\n
- Trigger messages match

- Network answers are different
Conclusion for this technique

- number and expressiveness of computed interactions
- rapidity of computations
- simplicity of use
- run on 23 services (FIW contests)
Plan

1. Our method for phone services
2. Current and future works: SIP
SIP and VoIP services

- Services deployed on devices, proxies, etc
- End systems are more “intelligent” than phones
- Work in progress at France Telecom
  - formalism to be adapted
  - algorithms to be adapted

--> Intellectual property issues...
CFBandAutoRedial

Bsip : CFB(Bsip’)

Aip.invite(Bsip, Asip, Bsip, Asip,(i, invite))

100trying(Asip, Bsip, (i, invite)).Aip

180ringing(Asip, Bsip, Bsip’, (i, invite)).Aip

Asip : AutoRedial

Aip.invite(Bsip, Asip, Bsip, Asip,(i, invite))

100trying(Asip, Bsip, (i, invite)).Aip

486busyhere(Asip, Bsip, (i, invite)).Aip

Aip.ack(Bsip, Asip, Bsip, (j, ack))

Asip.subscribe(Bsip, Asip, Bsip, Asip,(k, subscribe))
## Interactions (by hand)

<table>
<thead>
<tr>
<th>Configuration</th>
<th>Bsip : CBF(Bsip’)</th>
<th>Asip : AutoRedial</th>
</tr>
</thead>
<tbody>
<tr>
<td>Path</td>
<td>Aip.invite(...)</td>
<td>100trying(...).Aip</td>
</tr>
<tr>
<td>Trigger request</td>
<td>Aip.invite(...)</td>
<td></td>
</tr>
<tr>
<td>Conflicting answers</td>
<td>180ringing(...).Aip</td>
<td>486busyhere(...).Aip</td>
</tr>
</tbody>
</table>
Conclusion

- Graphic and formal specifications adapted to the domain
- Fully automatic method
  - automatic computation of problematic configurations
  - no integration
- Quality and number of found interactions
- Adaptation of formalism and algorithms : in progress