Rule-Based On-Line Feature Interaction Detection for IMS Call Control Services

Lina Ren, Fei Li, Qi Yu, Bo Yang
IBM China Research Lab
Beijing, China
E-mail: {renlina, yuqi, boyang}@cn.ibm.com

Abstract—This paper presents a rule-based on-line feature interaction detection approach for IMS call control services. We model dynamic service behaviors, intentions and party relationships in simple and effective service notations carried in extended SIP headers at run time. A small set of service actions is abstracted from possible service operations upon SIP messages. Based on the service behavior model, two groups of feature interaction detection rules, action interaction rules and address interaction rules, are developed to describe the condition when pairs of call control services interacts with each other. By applying the rules on appropriate detection set of service notations, conflicted service instances can be identified for following resolution. We present experiment results that proves the effectiveness of our approach.

I. INTRODUCTION

Originally specified by 3GPP, IMS (IP Multimedia Subsystem)[1] has become the infrastructure for future telecom network. With its well-defined architectural framework, IMS offers service providers the ability to create, provide and manage services in a fast and easy way. However, the flexible and distributed nature of IMS service provisioning also brings new challenges from the perspective of feature interaction, a traditional problem long existing in telecom network.

Feature interaction has long been recognized as an inherent problem in systems with multiple units of client-valued functionality working together but not aware of each other. Clearly, these functionalities may interfere with each other and result in unintended consequences. This is known as general feature interaction. During telecom services provisioning, numerous services may be independently developed, deployed and invoked to operate on the same call. This is a typical environment for feature interaction phenomenon to occur. Since the problem was identified and the term Feature was first coined by Bellcore at early 80’s, many different definitions have been proposed for Feature and Feature Interaction, for example, “Feature is the smallest part of a service that can be perceived by the service user”[2], and “Feature interaction occurs when one feature modifies or subverts the operation of another one” [3]. Many other definitions are discussed in [5][6]. Meanwhile, numerous approaches have been investigated to deal with the problem.

However, the problem has not been well addressed in new IMS service environment, where the problem gets worse, as will be discussed in detail in Section II. According to IMS design rationale, IMS services are not to be standardized, and many of them may be designed to meet temporary market needs, and may have short lifecycles, these are significant differences from traditional telecom network where limited standardized services are provided over long periods. So in IMS it is not realistic for telecom operators to handle feature interaction by manual inspection or exhaustive integrated testing, like they always did in the past. A general and practical approach is needed.

All these considerations motivate the design of rule-based on-line approach in this paper to detect feature interactions among SIP [4]-based IMS call control services at run-time. We set up a service behavior model to describe and organize all necessary information that should be considered when detecting feature interactions. In this model, the abstracted behaviors of a service instance, the relationship of involved specific user addresses, and the intentions of the service instance are all described. A concept of Super Dialog is created to define the range of service instances to be detected, and the detection is based on 8 rules abstracted from various typical existing services. The approach is implemented and validated in a feature interaction detector prototype.

This paper is organized as follows. Section II introduces the IMS service provisioning architecture and discusses feature interaction problem in new IMS environment. Section III overviews our feature interaction detection approach. Section IV describes the definition of service notations in detail, and section V establishes two groups of rules to detect feature interactions. Section VI presents the experiment results and discussions. Section VII reviews some related works. Finally, the paper concludes in Section VIII.

II. IMS SERVICE ARCHITECTURE AND FEATURE INTERACTION

A. The IMS Service Provisioning Architecture

In IMS, application services are deployed on different Application Servers (AS). The S-CSCF (Serving-Call Session Control Function) is responsible to trigger services and forward SIP messages to Application Servers in pre-defined order according to iFCs (Initial Filtering Criteria) downloaded from HSS (Home Subscriber Server). Then services in Application
Servers process received SIP messages based on service logic of itself, acting as a SIP UA, proxy, or B2BUA[4].

3GPP has proposed an entity called SCIM (service capability interaction manager [5]) to manage service interactions, and the feature interaction detection function discussed in this paper can be regarded as a functional component of SCIM.

![Functional architecture for IMS service provisioning](image)

Figure 1. Functional architecture for IMS service provisioning

**B. The Feature Interaction Problem in IMS**

The intrinsic complexity of feature interaction problem is well documented, and numerous researchers have investigated various techniques to deal with it. However, it has not been completely solved even in traditional telecom network [6] [7] [8]. In IMS environment, the problem gets even more complicated:

1) **Increasing service execution points.** With the almost unlimited flexibility provided by SIP, IMS services can be deployed and executed at many locations, like in proxy servers, redirect servers, and even user agents. Thus a central mechanism of control is more difficult to work;

2) **Increasing mechanisms of service control.** IMS service architecture allows three different service control mechanisms: SIP-based, OSA API-based and CAP-based, Web Services can also be leveraged in service control (like Parlay-X). These mechanisms all have their own understanding and approaches, but in IMS they can co-exist to work on the same call. Thus the interaction between their provisioned services is inevitable and challenging to resolve.

3) **New actors introduced in service creation.** With the trend to expose telecom network capabilities as open interfaces, 3rd party service providers, end users, web application developers are invited to create services. This trend surely brings more services, but also uncertainties in service behaviors, thus feature interaction are more likely to happen.

Hence, it is critical to revisit the feature interaction problems again in IMS environment.

**C. IMS Service Feature Interaction Examples**

Some typical IMS service feature interaction examples are as follows.

1) **Redirect by Location (RL) and Redirect by Presence (RP)**

Both Redirect by Location (RL) and Redirect by Presence (RP) services want to redirect calls to the most convenient terminal for service user. However, because the two services depend on different conditions, the redirect actions may interfere with each other. Assume Bob’s cell phone registered a RL service: When he is at office, redirect all calls to his office phone. If office phone is busy, then redirect to cell phone. The RL service can work well individually, but consider the situation when his office phone registered a RP service: If the office phone is busy, redirect the call to a voice mailbox. If someone calls Bob’s cell phone when Bob is at office, and the office phone is busy, the intended action of RL may be replaced by RP.

2) **Between Call with Background Music Services**

Call with Background Music (CBM) service provides users pre-subscribed background music when user is on a call. The feature interaction occurs when originating party and terminating party both subscribed CBM and with different pieces of music.

**III. APPROACH OVERVIEW**

In this section, we introduce our rule-based feature interaction detector in IMS environment.

**A. Problem Analysis**

“Feature interaction” is actually the incompatibility among related service features, or service behaviors. So to detect feature interaction at runtime, three sub-problems need to be solved: 1) How to model service behaviors appropriately, to express service features; and how to get and maintain such information at runtime. 2) How to define the feature interaction conditions and express the conditions as general feature interaction detection rules. 3) How to identify the set of service instances to put together to detect possible feature interaction, i.e. identifying the detection range. How we resolve these problems are briefly described as follows.

**B. Service Behavior Model**

In our service behavior model, we identified three types of behavior information related to call control service features: 1) Service Actions, like “replacing terminating addresses”, 2) Service Intentions, like the intention “do not receive call from Alice”, and 3) Party Relationships, like the originating, terminating and triggering party addresses. The behavior model is explained in detail in Section IV.

On how to get and maintain these behavior information for each IMS service instance, because only IMS services themselves are capable to provide the information about their own behaviors, services are supposed to explicitly add a “Service Notation” in extended SIP headers to express their behavior, when sending out SIP messages. Thus, service behaviors are actually service notations.

**C. Feature Interaction Detection Rules**

Two types of feature interaction detection rules are designed in Section V, to identify both 1) Action interaction and 2) Address interaction accordingly.
D. Feature Interaction Detection Range

When detector receives a new SIP message (with a service notation) from a service instance, it is time to determine the detection set of service notations, i.e. notations that should be put together to check if there is any feature interactions.

When routing a SIP request from originating party to terminating party, several service instances may be triggered. Each service instance may involve several users (addresses). All users involved in these triggered service instances form one Super Dialog. The concept of Super Dialog is illustrated in Figure 2. In this example, A calls B, B forwards the call to C and C forwards the call again to D, two call forwarding unconditional (CFU) service instances I and II are triggered. The Super Dialog for this call is user {A, B, C, D}. So at the moment when the call forwarding service of user C is triggered, the feature interaction detection range should be: all service instances active on users A, B, C and D.

So the feature interaction detection set is defined as: all service notations of active services of users belong to the same Super Dialog.

E. Rule-Based Feature Interaction Detector

Based on the discussion above, a rule-based feature interaction detector is illustrated in Figure 3. It is located between S-CSCF and AS, and talks to them via the SIP/ISC interface to detect possible feature interactions and report them to another interaction resolution entity. Within the range that service feature interactions should be detected by the detector, all SIP messages between S-CSCF and AS are supposed to traverse the detector. The detector is consisted of 3 modules: Notation Parser, Super Dialog Container and Rule-Based Detection.

1) Notation Parser. It extracts service notation in SIP messages received from AS, and forwards the message to Super Dialog Container.

2) Super Dialog Container (SDC). It manages all Super Dialogs and finds the detection set of service notations for any new service notation received from Notation Parser. The runtime relationships among Super Dialogs, Calls and Service Notations, are illustrated in Figure 4.

A Super Dialog may contain multiple SIP calls, and a call may have multiple service notations from several service instances triggered in this call. Upon receiving a SIP message, SDC identifies the users in the same Super Dialog, then the detection set of service notations based on these users, and sends the detection set of notations to Rule-Based Detection to check if there is any feature interactions. After that, SDC updates its data with the new notation if necessary (Some service behaviors are only effective once and need not to be stored in SDC, like CFU service replacing call terminating address, services can send different types of notations to indicate whether its behavior persists). Although a service instance may have several behaviors during a call, it is valid to assume that one service instance may only have at most one valid service notation at any time. So a new notation from the same service instance replaces the old one if there is.

For deletion of notations, service instances are required to explicitly include a Finish Indication notation in final SIP responses or Ack to final responses as soon as the call is successfully established or fails. Stored notations are deleted from SDC when service instance explicitly includes Finish Indication notation in SIP messages or when the SIP call of the notations terminates.

3) Rule-Based Detection. It is a rule engine which applies all feature interaction detection rules to pairs of notations in detection set, to look for interactions. If an interaction is found, it notifies a feature interaction resolution system to handle the detected interaction (The discussion of interaction resolution methods is out of the scope of this paper).
IV. SERVICE NOTATIONS

As discussed in Section III, service notations are supposed to be included in SIP messages sent out by service instances to express behaviors of service instances for feature interaction detection. This section focuses on the definition of service notations used in our approach.

A. Service Behavior Definition

From the perspective of feature interaction, a service can be regarded as a series of control behaviors. Let us denote behaviors by Bi, then a service S is:

\[ S = \langle B_1, B_2, ..., B_n \rangle. \]  

(1)

Services may take different actions toward three parties of a call—the originating (Orig) party, the terminating (Term) party and the triggering (Trig) party. Thus a behavior B is denoted as:

\[ B = \langle \text{Orig}, \text{Trig}, \text{Term} \rangle. \]  

(2)

For example, if a service is triggered at user A and makes a call from A to B, the Orig and Trig are same and both describe service control behavior toward A, and Term describes the service control behavior toward B. Sometimes when a behavior involves three parties of different users, like call forwarding services, Trig is different from Orig and Term.

Further, the behavior toward any particular party (Orig, Term, Trig), take Orig for example here, is defined as

\[ \text{Orig} = \langle \text{Addr}, \text{Act} \rangle. \]  

(3)

In this definition, Addr means “address”, it is defined as:

\[ \text{Addr} = \langle \text{PA, C} \rangle, \]  

(4)

where PA (party address) is the set of specific user addresses of this party in this call, and C (constraint) is the service constraints on party addresses. The constraint expresses service intentions on party addresses: which set of user addresses can be accepted as a specific party. For example, the constraint “Not A” in behavior toward Orig party means: the service does not allow user with address A to be the Orig party. Addr is used in detecting address interactions in Section V.

Act means “action”; it is used in detecting action interactions in Section V. The actions are what services do to particular parts of a SIP message, but operations to address-related parts of message is not regarded as actions. Act is defined as:

\[ \text{Act} = \{ \text{Add} (\text{Field}, \text{Para}), \text{Delete} (\text{Field}, \text{Para}), \text{Modify} (\text{Field}, \text{Para}) \}. \]  

(5)

This definition of service actions is based on the fact that all SIP-based service actions are achieved by adding, deleting or modifying some certain parts of SIP messages. Although we can also extract actions from existing services, by analyzing their large amount of various actions, it is difficult to give a general and exhaustive set of all possible actions. So for its simplicity and expressiveness, we describe actions with operations applied to SIP messages. Field indicates which type of SIP message part that the operation affects; it is defined as

\[ \text{Field} = \{ \text{Header, Body, Media} \}. \]  

(6)

Para denotes the specific changed part in Field, and the set can be freely defined according to SIP message formats. For example, some Para for changed headers can be Privacy, Request-Disposition, and P-Called-Party-ID. And Para for contents can be presence, IM (instant message), register.

B. Examples Using Behavior Definition

The above definitions of service behaviors are used to generate service notations. Several examples are as follows, with service notations expressed as tables for conveniences. The element on the upper left corner in tables is Service-ID to identify the behavior is from which service instance. Unused fields is denoted as “.”.

1) Closed User Group (CUG) service.

The service restricts the contact range of users. A user in a pre-configured group could only communicate with users in the same group; in this example the group includes addresses A, B and C. The service behavior is described in Table 1.

<table>
<thead>
<tr>
<th>Table 1.1</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLOSED USER GROUP OF USER A, B, C, OUTGOING CALL BEHAVIOR</td>
</tr>
<tr>
<td>( \text{SI} ) &amp; ( \text{Orig} ) &amp; ( \text{Trig} ) &amp; ( \text{Term} )</td>
</tr>
<tr>
<td>Addr &amp; C &amp; PA &amp; C &amp; PA &amp; C &amp; PA</td>
</tr>
<tr>
<td>Act &amp; - &amp; - &amp; - &amp; - &amp; -</td>
</tr>
</tbody>
</table>

2) Originating Call Screening (OCS) Service

In this example, the service restricts a user A from making call to B and C. The service behavior is described in Table 2.

<table>
<thead>
<tr>
<th>Table 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>OUTGOING CALL SCREENING BEHAVIOR</td>
</tr>
<tr>
<td>( \text{SI} ) &amp; ( \text{Orig} ) &amp; ( \text{Trig} ) &amp; ( \text{Term} )</td>
</tr>
<tr>
<td>Addr &amp; C &amp; PA &amp; C &amp; PA &amp; C &amp; PA</td>
</tr>
<tr>
<td>A,B,C &amp; B &amp; - &amp; A &amp; - &amp; A</td>
</tr>
<tr>
<td>Act &amp; - &amp; - &amp; - &amp; - &amp; -</td>
</tr>
</tbody>
</table>

3) Call Forwarding on Unconditional (CFU) Service

In this example, the service forward all calls to B to another user address, C. The service behavior is described in Table 3.

<table>
<thead>
<tr>
<th>Table 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>CALL FORWARDING UNCONDITIONAL BEHAVIOR</td>
</tr>
<tr>
<td>( \text{SI} ) &amp; ( \text{Orig} ) &amp; ( \text{Trig} ) &amp; ( \text{Term} )</td>
</tr>
<tr>
<td>Addr &amp; C &amp; PA &amp; C &amp; PA &amp; C &amp; PA</td>
</tr>
<tr>
<td>- &amp; A &amp; - &amp; B &amp; - &amp; C</td>
</tr>
<tr>
<td>Act &amp; - &amp; - &amp; - &amp; - &amp; -</td>
</tr>
</tbody>
</table>

4) Calling Line Identification Presentation (CLIP)

The service requires originating party to provide its line identification. The service behavior is described by Table 4.
1. This is because operations on address-related parts of SIP messages. This definition of actions greatly simplifies the parameters actions toward SIP messages, and actions have same types of effective and can be deleted.

In addition to the notations that describe service behaviors, another special notation, Finish indication, should also be included in SIP messages under certain conditions as discussed in Section III, to notify the detector the notation is no longer included in SIP messages under certain conditions as discussed in another special notation, Rule 1 is formalized as:

\[
\text{Act} = \begin{cases} 
\text{Add(Head, privacy)} & \text{if the service adds a privacy header to SIP message, the Act is filled with Add(Head, privacy).} \\
\text{B} & \text{if another special notation is filled} \\
\text{A} & \text{if another special notation is not filled} \\
\text{-} & \text{if there is no action} 
\end{cases}
\]

Because the service adds a privacy header to SIP message, the Act is filled with Add(Head, privacy).

V. FEATURE INTERACTION DETECTION RULES

This section describes a set of rules developed to detect feature interactions among IMS call control services. The detection rules describe the conditions when feature interaction between services happens, and these conditions all trigger the same corresponding action of detector to report that a feature interaction problem has been detected. The rules are defined upon service notations, and are applied to each detection set of service notations.

We group all feature interactions in IMS call control services into two basic types: 1) Action Interaction, which occurs when multiple services intend to work on the same part of a same SIP message, and 2) Address Interaction, which occurs when assumptions on party addresses are violated.

Accordingly, two types of interaction rules are designed, which is action interaction rule and address interaction rule. Action interaction rules are checked against service notation pairs in detection set, by evaluating the compatibility of their actions. Address interaction rules are also checked by comparing addresses in notation pairs.

A. Action Interaction Rule

As discussed in Section IV, there are only three types of actions toward SIP messages, and actions have same types of parameters (Field, Para) for they all operate on some parts in SIP messages. This definition of actions greatly simplifies the action interaction rule.

**Rule1:** Interaction happens between services with actions on same part of a same SIP message, with the exception of address-related parts.

Let P1 be any party of a service notation, and P2 be any party of another service notation, Rule 1 is formalized as:

\[
(P_1.\text{Addr.PA} \cap P_2.\text{Addr.PA} \neq \emptyset) \\
\land (P_1.\text{Act.Field} = P_2.\text{Act.Field}) \\
\land (P_1.\text{Act.Para} = P_2.\text{Act.Para})
\]

It should be noted that Rule 1 only applies to those parts of SIP messages which are not related to party addresses. For example, the From and To headers are out of the scope of Rule 1. This is because operations on address-related parts of message are common and not necessarily incur feature interaction, and the address related interactions are handled in following Address Interaction Rules. So as explained in Section IV, the operation to address-related parts of a SIP message, like From header, are not regarded as actions.

It should also be noted that the service notations that should be detected against Rule 1 is not the whole detection set given by SDC, but actually a subset of it: only those service notations belong to a same Super Dialog, and with the same SIP method name. This is to make sure Rule 1 works on “same part of a same SIP message”.

Many services are implemented by changing the message contents—Field and Para, There are abundant cases of this kind of service interactions, for example, the interaction between CLIP and CLIR (Calling Line Identification Restriction) services. Another example is the conflict between different back ground music among multiple CBM services.

B. Address Interaction Rules

Address interaction rules are more complicated than the action rules because a Super Dialog may contain many directly or indirectly related sessions. We categorize the address interaction rules into three classes: Address Incompatible, Forwarding Loop and Attempt Conflict.

1) Address Incompatible

The address incompatible condition of feature interaction means a call attempt is attempted one service but not allowed by another service. Such situations often happen when involving services which limit user’s contact range as OCS (Outgoing Call Screening), ICB (Incoming Call Barring) and CUG (Closed User Group).

**Rule2:** Interaction happens between services with incompatible originating party addresses:

\[
(\text{Term}, \text{Addr.PA}) \cap (\text{Term}, \text{Addr.PA} \neq \emptyset) \\
\land (\text{Orig}, \text{Addr.PA} \neq \text{Orig}, \text{Addr.C}) \\
\land (\text{Orig}, \text{Addr.PA} \neq \text{Orig}, \text{Addr.C})
\]

**Rule3:** Interaction happens between services with incompatible terminating addresses:

\[
(\text{Orig}, \text{Addr.PA}) \cap (\text{Orig}, \text{Addr.PA} \neq \emptyset) \\
\land (\text{Term}, \text{Addr.PA} \neq \text{Term}, \text{Addr.C}) \\
\land (\text{Term}, \text{Addr.PA} \neq \text{Term}, \text{Addr.C})
\]

2) Forwarding Loop

This type of rules describes interaction conditions between forwarding services with a loop in party addresses. Various call forwarding services may forward calls according to time, presence, location, etc. and produce many interactions, because they change the initial user request.

**Rule4:** Interaction happens between services triggered at different users to call each other.
particular user triggered two forwarding services at different times. To forward a same call establishing request to different users, consecutive forwarding services may cause undesired consequences. Interactions can be detected after notation combination.

Rule 3. This is just one typical example, many other action interactions can be detected after notation combination. After new notation is formed, all rules should be checked against the new notation in detection set.

The typical example to apply this combination rule is call screening after two forwarding services: Consider the situation with multiple users and multiple sessions. Possible interactions to validate our detection approach. Table 5 lists the services we use and interactions detected with rule numbers matched. Our model is able to handle complex situations with multiple users and multiple sessions.

3) Attempt Conflict

The attempt conflict rules apply for interactions in forwarding services which cause undesired consequences.

Rule 7: Interaction happens when two forwarding services want to forward a same call establishing request to different users.

\[(\text{Orig}_1, \text{Addr}.PA \neq \text{Term}_1, \text{Addr}.PA \neq \emptyset)\]
\[\wedge (\text{Term}_1, \text{Addr}.PA \cap \text{Trig}_2, \text{Addr}.PA \neq \emptyset)\]

Rule 8: Interaction happens when calls requests from one particular user triggered two forwarding services at different addresses, but forwarded to a same destination address. This is possible in SIP services because SIP provides the capability to fork multiple requests.

\[(\text{Orig}_1, \text{Addr}.PA \neq \text{Trig}_2, \text{Addr}.PA \neq \text{Term}_2, \text{Addr}.PA)\]
\[\wedge (\text{Orig}_2, \text{Addr}.PA \neq \text{Trig}_2, \text{Addr}.PA \neq \text{Term}_2, \text{Addr}.PA)\]
\[\wedge (\text{Trig}_2, \text{Addr}.PA \neq \text{Trig}_2, \text{Addr}.PA \neq \emptyset)\]
\[\wedge (\text{Term}_1, \text{Addr}.PA \cap \text{Term}_2, \text{Addr}.PA \neq \emptyset)\]

VI. EXPERIMENTS AND DISCUSSION

A. System Implementation

In following experiments, the feature interaction detector and call control services are implemented as SIP Servlets following JSR 116[9] standard, and deployed on WebSphere Application Server[10]v6.1 environment. We also implement an S-CSCF simulator capable of triggering SIP request to AS and routing messages. We use DROOLS[11] as our rule engine. Call control services in following experiments add service notation in outgoing SIP messages with extended SIP headers to describe its current behaviors. The detector uses this information to look for interactions, before the service really affects other network entities. The service notation is carried by four SIP headers extended for feature interaction detection. For example, the outgoing message of CFU is:

```
INVITE userC@ibm.com SIP/2.0
From: "userA"<sip:userA@ibm.com>
To: "userB"<sip:userB@ibm.com>
Service-Id: 1
Notation-orig: addr.C=-; addr.PA=userA@ibm.com; act=-
Notation-trig: addr.C=-; addr.PA=userB@ibm.com; act=-
```

B. Experiments

We choose a set of typical and popular services with potential interactions to validate our detection approach. Table 5 lists the services we use and interactions detected with rule numbers matched. Our model is able to handle complex situations with multiple users and multiple sessions.

```
<table>
<thead>
<tr>
<th></th>
<th>CFX</th>
<th>OCS</th>
<th>AR</th>
<th>CUG</th>
<th>MP</th>
<th>CLIP</th>
<th>CLIR</th>
<th>CBM</th>
</tr>
</thead>
<tbody>
<tr>
<td>CFX</td>
<td>4,5,7,8</td>
<td>3</td>
<td>2,3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OCS</td>
<td></td>
<td></td>
<td></td>
<td>2,3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CUG</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2,3</td>
<td></td>
</tr>
<tr>
<td>MP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CLIP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CLIR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>CBM</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

CFX: Any kind of call forwarding services
AR: Automatic Ringback
MP: MultiParty Call
C. Discussion

In this part, we discuss some design options and their impact on detection result.

The abstraction level. Obviously, our service behavior model of notations is partial and not capable of capturing all details of service behavior, e.g. conditions to take the behaviors. So we cannot describe service behaviors accurately at feature interaction detector, and some services may have a same behavior description in our model. At the same time, one service may have different runtime behaviors under different conditions. However, this imperfect description of service behaviors would not affect the ability of the detector to identify feature interactions. Also the great challenges to list all these condition description exhaustively make it nearly impossible to predict what kind of condition would be available in the future. So the detector focuses on runtime behavior of certain service instances, no matter what service it is and what the service intends to achieve on the high level.

Special users. This problem happens when some of the user involved is not “real user”, as voice mailbox. From the perspective of service behaviors, connect a call to a voice mailbox or a real user has no difference. But the voice mail has impact on some services for it would answer at any circumstance, and real users would not. For example, Simultaneous Ringing service would be interfered if one of its contacted addresses has voice mail service. Same case is with Sequential Ringing service. This problem could be solved by adding a special address, e.g. vm, to our address set in behavior model. The cost is to lose generality of our model.

VII. RELATED WORK

The feature interaction problem in telecom network has been studied for a long time and many pioneer works addressed this issue [12][13]. Some work has been carried out on feature interactions in SIP architecture. Chan and Bochmann [14] proposed an off-line approach to reduce feature interaction problems in SIP. They derived a formal SDL (Specification and Description Language) model from SIP specification. A set of methodology was presented to detect and prevent feature interactions in SIP. Kolberg and Magill [15] handled SIP call control services by rules too. They modeled service behaviors by connection before and after the service triggering. They used a specific “Treat” party to express network operation. By deploying the approach on proxies, they could detect distributed service interactions. Their rules were effective for some interactions among call control services but did not capture specific properties and actions in SIP. Blair and Turner [16] applied their policy architecture to SIP services. They proposed a policy-based approach to describe services and detect service interactions as policy conflicts. But their approach is based on non-standard service provisioning approach which would be a major obstacle for industrial application.

VIII. CONCLUSIONS

In this paper we have proposed a rule-based approach to detect IMS feature interaction at run time. By defining service notations, we achieved a simple but expressive run time service behavior model. We use 2 groups of rules to detect feature interactions, the action interaction rule and address interaction rules. The action interaction rule covered a large amount of interactions, for our action definition is abstracted from general operations to SIP messages. The address rules covered complex party and session relationships in IMS call control services including consecutive forwarding cases. The effectiveness and generality of the proposed approach has been proved in experiments conducted among typical call control services. More rules will be defined in the future to handle more complex situations and cover IMS services not related to call control.

REFERENCES