Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X

IJCSMC, Vol. 2, Issue. 12, December 2013, pg.55 – 63

RESEARCH ARTICLE

Policy Based Web Service Interaction Management

Evelina Pencheva

Faculty of Telecommunications, Technical University of Sofia, Bulgaria

enp@tu-sofia.bg

Abstract—Service interaction manifest itself as an unexpected behaviour of multiple services used together either simultaneously or in sequence. The paper describes a policy based approach to web service interaction management. Parlay X web services for call control, mobility, user interaction and policy management are studied. Web service behaviour is described as a set of policy rules. Rules are generated using descriptive logic. Web service interaction is considered as a satisfiability problem. The policy evaluation uses a standard inference algorithm to detect web service interactions.

Keywords— Parlay X; Policy management; Service interaction; Descriptive logic; Inference algorithm

I. INTRODUCTION

The technology of Parlay X Web Services provides open access to communication network functions. Third party applications can use standardized application programming interfaces (APIs) to control mobility, messaging, call and session processing, charging etc. Parlay X APIs are simple and easy to use. They provide a high level abstraction of communication functions and thus stimulate service development. As the number of web services increases with API use, interest in service interworking also increases. Service interworking refers to multiple services used together either simultaneously or in sequence. Although service interworking usually causes no problem, in some cases the result is not what it is expected. Service interaction occurs as an unexpected behaviour due to direct conflict between service requirements or to failure in meeting service requirements by combined execution of multiple service logics [1], [2], [3].

Service interaction problem has been intensively studied in academia and industry [4]-[9]. Service interactions may occur between various types of services, ranging from single-transmission bilateral/multilateral interactions, multi-transmission interactions and routing interactions [10]. The problem has been widely investigated for services, related to call control and messaging. However, sufficient attention has not been paid to interactions caused by mobility management, e.g. interactions that depend on mobile terminal status and location, as well as user interactions, e.g. playing text, voice, audio messages.

Service interaction management is aimed mainly to detection and resolution. Service interaction is typically detected during specification time using formal models [11], [12]. When a conflict between services occurs in the network, it has to be solved. Third Generation Partnership Project (3GPP) defines Service Capability Interaction Manager as a special type of Application Server (AS) which performs the role of interaction management between other application servers, but its functionality is not specified [13].

This paper presents a policy-based approach to describe Parlay X web services and to manage web service interactions by policy evaluation. Parlay X “Policy” web service interfaces are used for policy provisioning and

evaluating. Web service functionality is formally described and presented as a set of policy rules. The service behaviour is considered as satisfiability problem and service interaction is detected automatically by using standard inference algorithm. Service interactions may be reported as events in the process of policy evaluation. The approach applicability is illustrated in the context of Parlay X "Third party call", "Audio call", "Terminal status" and "Terminal location" web services. The approach may be extended easily to quality of service control, messaging and multimedia broadcasting.

The paper is structured as follows. Section II provides a short introduction to descriptive logic. Section III presents the basic functionality of Parlay X "Policy" interfaces. In Section IV, Parlay X web services for call control, user interaction, and mobility are presented. Section V gives formal descriptions of Parlay X application view on call and user interaction control provided by "Third party call" and "Audio Call" web services, and mobility management provided by "Terminal status" and "Terminal location" web services. Parlay X applications with value-added logic are defined by refinement of mobility management and call control in Section VI. Section VII illustrates the usage of standard inference algorithm to detect service interaction and applicability of Parlay X "Policy" interfaces for policy evaluation. In the last section, conclusions are discussed and the future work is outlined.

II. BASICS OF DESCRIPTIVE LOGIC

Having a domain Δ with fixed names of the concepts and the roles, and given constants form a triplet $\langle C, R, A\rangle$ thus that one can define the set of the concepts C, terminologies T and assertions A (i.e. allowed formulae in TBox and ABox respectively) as follows:

$$\begin{aligned} BC &:= \top \mid C \mid \neg BC \mid BC \sqcap BC \\ C &:= BC \mid \forall R.BC \mid \exists R.BC \\ T &:= BC \sqsubseteq C \mid BC \equiv C \\ A &:= a:C \mid (a,b):R \end{aligned}$$

where in C is any basic concept *C*, in R is any basic relation in *R*, and in A are constants a, b. The pair $\langle T, A\rangle$ is *knowledge base* where $T \subseteq T$ and $A \subseteq A$.

Interpretations of description logics are $I = (\Delta^I, \cdot^I)$ where Δ^I is non-empty set and $\cdot^I$ is mapping of subsets of Δ^I onto the concept names, relations over Δ^I onto role names, and elements of Δ^I onto constants. The interpretations are extended over C as follows:

$$\begin{aligned} (\top)^I &= \Delta^I \\ (\bot)^I &= \varnothing \\ (C \sqcap D)^I &= C^I \cap D^I \\ (\neg C)^I &= \Delta^I \setminus C^I \\ (\forall R.C)^I &= \{ a \in \Delta^I \mid \forall b\, ((a,b) \in R^I \rightarrow b \in C^I) \} \\ (\exists R.C)^I &= \{ a \in \Delta^I \mid \exists b\, ((a,b) \in R^I \wedge b \in C^I) \} \end{aligned}$$

In fact, $\exists R.C$ means the whole subset of elements of Δ that are in relation *R* with the element *C*. The duality of operator $\forall$ can be expressed in terms of operator $\exists$, as $\forall R.C = \neg\exists R.\neg C$.

The definition of *satisfaction* is intuitive as a relation between interpretations and terminologies or assertions. Thus $\models$ is relation between *I* and all formulae supported by *I*:

$$\begin{aligned} I &\models C \sqsubseteq D && \text{iff } C^I \subseteq D^I \\ I &\models C \doteq D && \text{iff } C^I = D^I \\ I &\models a:C && \text{iff } a^I \in C^I \\ I &\models (a,b):R && \text{iff } (a^I, b^I) \in R^I \end{aligned}$$

So, for a subset $K \subseteq T \cup A$ one may state that $I \models K$ iff $I \models \varphi$: $\forall \varphi \in K$. In general, consequence is if having a knowledge base $\langle T, A\rangle$ and formula φ such that $\varphi \in T \cup A$ then φ follows from $\langle T, A\rangle$ i.e. $\langle T, A\rangle \models \varphi$ iff $\forall I$: $I \models \langle T,A\rangle \Rightarrow I \models \varphi$ where $\Rightarrow$ notes implication. One of the main reasoning tasks in description logics is to check if given formula follows from given knowledge base.

III. PARLAY X INTERFACES FOR POLICY MANAGEMENT

The Parlay X "Policy" web service provides simple means for applications to make use policies by policy provisioning and policy evaluation [14]. In the context of the service, policy is an ordered combination of policy rules, defined to manage and control the access to network resources. Policy rule is a combination of conditions and actions to be performed if the condition is true. Rules may be grouped in specific domains. Policy action is associated to a policy condition in a policy rule and it is executed when the associated policy condition results in "true" from the policy evaluation step. A condition is a Boolean predicate which value is true or false. Policy evaluation is the process of evaluating the policy conditions and executing the associated policy actions up to the point that the end of the policy is reached.

PolicyProvisioning interface includes the operation to create, modify, view, and delete policies. The interface provides methods for specified policy domain management (creation, query, deletion), for rule management (creation, modification, deletion, and query). Fig.1 shows the sequence diagram of querying a rule list and modifying a rule.

PolicyEvaluation interface provides operations to request evaluation of policies.

PolicyEventNotificationManager interface is used to manage the application subscription to receive notifications about policy related events. PolicyEventNotification provides functions for delivering the notification to the application when the event occurs.

.

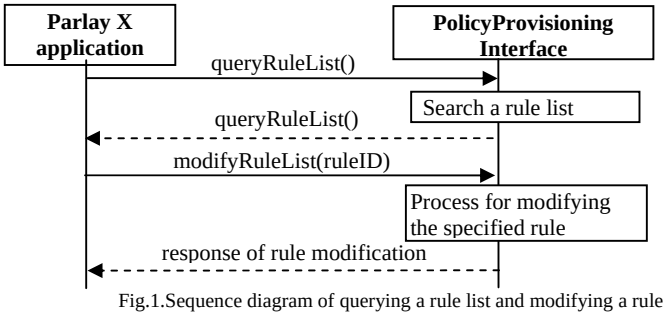

Fig.1.Sequence diagram of querying a rule list and modifying a rule

IV. CALL AND USER INTERACTION CONTROL AND MOBILITY MANAGEMENT THROUGH WEB SERVICES

The Parlay X defines several Web Services that allow third party application to control call establishment and treatment.

The "Third party call" Web Service provides functions to applications for creating and managing a call initiated by an application (third party call) [15]. An application setting up of a call session creates a "context" that represents a call session with usually two participants, or at a minimum one participant connected, a unique identifier is assigned to the call session. Subsequently, the application may wish to add, remove, park or transfer call participants. It is also possible to retrieve the media details on call participants. The application can also force the call session and all its participants to be terminated.

The "Audio call" Web service provides a flexible way to provide multimedia message delivery and the dynamic management of the media involved for the call participants [16]. The interface is very simple, not requiring the developer to manage the creation of the call. The "Audio Call" Web service allows media to be added/dropped for any ongoing call. This Web service also allows interaction with other call control Web services (e.g. multimedia conference, third party call), enabling delivery of multimedia to call participants in an ongoing call.

Two Parlay X Web Services are defined in order to allow access to mobility management functions in the network.

The "Terminal status" Web service provides access to the status of a terminal or a group of terminals through requests or notifications of a change in the status [17]. The status of a terminal can be expressed as reachable, unreachable or busy.

The "Terminal location" Web service provides access to the location of a terminal through: request for the location of a terminal; request for the location of a group of terminals; notification of a change in the location of a terminal; notification of terminal location on a periodic basis [18]. Location is expressed through a latitude, longitude, altitude and accuracy.

V. FORMAL DESCRIPTION OF WEB SERVICE FUNCTIONALITY

This section provides a simplified formal description of Parlay X Web services, illustrating the basic idea.

The information in the knowledge base is split into ABox and TBox. The ABox contains assertional information (facts concerning individuals in the domain), and TBox contains terminological knowledge (definitions of derived notions) represented by formulas.

Atomic concepts represent the user state in the network. There are also states representing the application view on the call and user interaction in the context of Parlay X services. For the services described in the previous section, the following concepts are defined:

- $idle_A$ – the user A is idle
- $busy_A$ – the user A is busy
- $unreachable_A$ –the user A is not reachable
- $notifiedIdle_A$ – the application is notified that A is idle
- $notifiedBusy_A$ – the application is notified that A is busy
- $notifiedUnreachable_A$ – the application is notified that A is unreachable
- $3rdPartyCall_A$ – there is a third party call to A.

The atomic concepts representing the user state may be defined as policy conditions in terms of Parlay X "Policy" web service.

The atomic roles represent the Parlay X Web service operations that modify the user state. The following atomic roles are defined:

- $makeCallSession_A$ – a third party call is setup to A by a third party application.
- $endCallSession_A$ – the third party call to A is disconnected by the application.
- $playTextMessage_A$ – a text message is played to user A.
- $playAudioMessage_A$ – an audio message is played to user A.
- $playVideoMessage_A$ – a video message is played to user A.
- $endMessage_A$ – the application cancels the message playing to user A.
- $getStatus_A$ – the application retrieves the A's terminal status .
- $startNotification_A$ – notifications of A's terminal status changes are made available to the application.
- $endNotification_A$ – notifications of A's terminal status are ended by the application.
- $statusNotificationUnreachable_A$ – a notification is received by the application that the subscriber A becomes unreachable.
- $statusNotificationIdle_A$ – a notification is received by the application that the subscriber A becomes idle.
- $statusNotificationBusy_A$ –a notification is received by the application that the subscriber A becomes busy.
- $getLocation_A$ – the application retrieves the location of A's terminal.
- $startGeographicalNofitication_A$ –notifications of A's terminal location changes are made available to the application.
- $startPeriodicNotification_A$ – -the application starts periodic notification of A's terminal location.
- $endNotification_A$ – -the application terminates notifications of A's terminal location.
- locationNotificationA – a notification about A's terminal location change is received by the application.

The atomic roles representing the web service operations may be defined as policy actions in terms of Parlay X "Policy" web service.

In the TBox, there are expressions that connect observable states of the users with the ones representing the application view. For simplicity, not all of the statements are presented:

- $3rdPartyCall_A \sqsubseteq busy_A$– if there is a third party call to A, then A is busy.
- $busy_A \sqsubseteq 3rdPartyCall_A$ – if the user A is busy, then she may be involved in a third party call.

The Tbox contains statements that express the relationship between the call and mobility management states:

- $\neg unreachable_A \sqsubseteq busy_A \sqcup idle_A$– if the user A is reachable then she is idle or busy
- $\neg idle_A \sqsubseteq unreachable_A \sqcup busy_A$ – if the user A is not idle then she is unreachable or busy.
- $\neg 3rdPartyCall_A \sqsubseteq unreachable_A \sqcup busy_A$ – if a third party call to user A can not be established, A is unreachable or busy.

Next, there are statements that specify how a user in the network and the application can change state using the "Third party call" and "Audio call" interfaces. For simplicity, just third party calls with one participant are considered:

- $idle_A \sqsubseteq \exists makeCallSession_A.3rdPartyCall_A$ – if the user A is idle then a third party call may be set up to A.
- $\neg idle_A \sqsubseteq \exists makeCallSession_A.(\neg\ 3rdPartyCall_A)$ – if user A is not idle then the call establishment to A fails.

- $3rdPartyCall_A \sqsubseteq \exists endCallSession_A.idle_A$ – if user A is involved in a third party call, then the application can end the call.
- $3rdPartyCall_A \sqsubseteq \exists playAudioMessage.3rdPartyCall_A$ – an audio message is played to A as a third party call participant. Similarly, while A is involved in a third party call, text and video messages may be played to A.

There are also statements that specify how a user in the network and the application can change state using the "Terminal location" and "Terminal status" interfaces. For simplicity, statements related to subscription to notifications are omitted and it is assumed that there are subscriptions to terminal status changes and terminal location changes:

- $unreachable_A \sqsubseteq \exists statusNotificationIdle_A.idle_A$ – if user A is unreachable, then she can attach to the network and become idle.
- $idle_A \sqsubseteq \exists statusNotificationUnreachable_A.unreachable_A$ – if user A is idle, then she can detach from the network and become unreachable.
- $idle_A \sqsubseteq \exists statusNotificationBusy_A.busy_A$ - if user A is idle, then she can be involved in a call and become busy.
- $busy_A \sqsubseteq \exists statusNotificationIdle_A.idle_A$ - - if user A is idle, then she can be involved in a call and become busy.
- $unreachable_A \sqsubseteq \exists getLocation_A.unreachable_A$ – while the user A is unreachable, then she can attach to the network and become idle.
- $idle_A \sqsubseteq \exists locationNotification_A.idle_A$ – while the A' terminal is idle, then it can change its location.
- $busy_A \sqsubseteq \exists locationNotification_A.busy_A$ – while the A' terminal is busy, then it can change its location.

Let us denote with USERs the set of all subscribers. By 3PCMM we denote the states s_i related to call control and mobility management. The Abox contains one statement presenting the initial state for each user: s_0: $\sqcap_{A \in USERS}(unreachable_A)$.

To express the fact that each user is in exactly one 3PCMM state at any moment the following statement is used:

$$\top \sqsubseteq \neg(\sqcup_{s1,s2 \in 3PCMM,\ s1 \neq s2}(s1 \sqcap s2)) \sqcap (\sqcup_{s \in 3PCMM} s)$$

The user state changes by means of actions defined as action functions.

The same is applied to the mobility management model where an action function $Func_{3PCMM}$ for given state corresponds to the possible transitions in the mobility management model. For example, the activation functions $Func_{3PCMM}(idle_A)=\{statusNotificationBusy_A\} \cup \{statusNotificationUnreachable_A\}$ describe all possible changes of $idle_A$ state. The fact that each user can change the 3PCMM state only by means of certain actions is represented by the following statement: for all $A \in USERS$, $s \in 3PCMM$ and all $R \notin Func_{3PCMM}(s)$, $s \sqsubseteq \forall R.s$.

All the statement may be defined as policy rules in terms of Parlay X "Policy" web service.

VI. Refinement of Parlay X Application Description

Services are modeled by refinement. The definition of refinement is formalized as refinement operation δ_F, for given service F, which operation transforms given knowledge base K into another knowledge base $\delta_F(K)$. The last is augmented by a set of activation concepts which generally are $A_F \subseteq \{ F_u \mid u \in SUB \}$. So, let $N \subseteq A_F$ then K and F interact on activation N if $\delta_F(K) \cup \{\top \sqsubseteq F_u \mid F_u \in N\} \cup \{\top \sqsubseteq \neg F_u \mid F_u \in A_F \backslash N\} \models \neg\top$. Let one has different services i.e. F_a, F_b : $a \neq b$. Then $\delta_{Fb}(\delta_{Fa}(K)) \equiv F_a \circ F_b$ and the services under consideration are such that $F_a \circ F_b \equiv F_b \circ F_a$. Contexts $C[\varphi]$ are used to define refinements in the knowledge base, where φ is a subformula of any formula ψ.

A. Call Forwarding Unconditional

The Call Forwarding Unconditional (CFU) service allows the called user to forward unconditionally her calls to another user. The refinement for CFU service is defined by the following statements:

$C_1[\neg \sqcup_{B \in SUSERS} CFU_{AB} \sqcap idle_A] \sqsubseteq \exists makeCallSession_A.C_2[3rdPartyCall_A]$ – if CFU is not activated from A to B and the user A is idle, then when the application initiates a third party call to A the call is established to A.

$C_1[CFU_{AB} \sqcap idle_A \sqcap idle_C] \sqsubseteq \exists makeCallSession_A.C_2[3rdPartyCall_B]$ – if CFU is activated from A to B and the user A is idle, then when the application initiates a third party call to A the call is established to B.

$C_3[\neg \sqcup_{C \in USERS} CFU_{AB} \sqcap \neg idle_A] \sqsubseteq \exists makeCallSession_A.C_4[\neg 3rdPartyCall_A]$ – if CFU is not activated from A to B and the user A is not idle, then when the application initiates a third party call to A, the call fails.

$C_3[CFU_{AB} \sqcap idle_A \sqcap \neg idle_B] \sqsubseteq \exists makeCallSession_A.C_4[\neg 3rdPartyCall_B]$ - – if CFU is activated from A to B and the user B is not idle, then when the application initiates a third party call to A the call, the call fails.

B. Greetings on Attachment

The Greetings on Attachment (GOA) service notifies the application that the user B attaches to the network and becomes idle. The application in turn initiates a third party call to B and it sends her a greeting message. The refinement for GOA service is defined by the following statements:

$C_1[\neg \sqcup_{A \in USERS} GOA_A \sqcap unreachable_A] \sqsubseteq \exists statusNotificationIdle_A.C_2(idle_A \sqcap \neg notifiedIdle_A)$ – if the GOA application is not activated and the user B is unreachable, when the user B becomes idle, the application is not notified.

$C_1[GOA_A \sqcap unreachable_B] \sqsubseteq \exists statusNotificationIdle_A.C_2(idle_A \sqcap notifiedIdle_A)$ – if the GOA service is activated and the user B is unreachable, then when the user B becomes idle, the application is notified.

$C_3(idle_A \sqcap notifiedIdle_A) \sqsubseteq \exists makeCallSession_A.C_4(3rdPartyCall_A)$ - if the user A is idle and the application is notified that the user A is idle, then a third party call is established to A.

$C_4(3rdPartyCall_A) \sqsubseteq \exists playAudioMessage_A.C_5(3rdPartyCall_A)$ – as a third party call the user A is played a greeting message.

A possible service interaction occurs when the user A which is unreachable, has forwarded her calls to user B which is also unreachable. When the user A attaches to the network, the GOA application is notified. Then the GOA application places a third party call to A, the call is forwarded to B which is unreachable.

C. Play Media in Forbidden Area

The Play Media in Forbidden Area (PMF) service notifies a user that she has entered in a forbidden zone. The service may be useful for parents that want to monitor the location of their children. For the sake of the service, new $inArea_A$ state is defined. When the user is $inArea_A$ state, the application is informed that the user is in the permitted area. The refinement for PMF service is defined by the following statements:

$idle_A \sqsubseteq \exists locationNotification_A.(inArea_A \sqcap idle_A)$ – while the user A is idle, the application is notified that A has changed her location and A is in the permitted area.

$busy_A \sqsubseteq \exists locationNotification_A(inArea_A \sqcap busy_A)$ – while the user A is busy, the application is notified that A has changed her location and A is in the permitted area.

$idle_A \sqsubseteq \exists locationNotification_A.(\neg inArea_A \sqcap idle_A))$ - while the user A is idle, the application is notified that A has changed her location and A is in the forbidden area.

$busy_A \sqsubseteq \exists locationNotification_A.(\neg inArea_A \sqcap busy_A))$ - while the user A is busy, the application is notified that A has changed her location and A is in the forbidden area.

$inArea_A \sqsubseteq idle_A \sqcup busy_A$ – while in the permitted area, the user A may be idle or busy.

$\neg inArea_A \sqsubseteq idle_A \sqcup busy_A$ - while in the forbidden area, the user A may be idle or busy.

$C_1[PMF_A \sqcap idle_A \sqcap (\neg inArea_A)] \sqsubseteq \exists makeCallSession_A.C_2[3rdPartyCall_A] \sqcap \exists playAudioMessage_A.C_2[3rdPartyCall_A]$ - when PMF is activated for user A and A is in forbidden area, then the application initiates a third party call to A and plays her an audio message. This service description is simplified. In fact, the application first has to retrieve the A's terminal status. If the user A is busy, then the application subscribes to receive notifications about A's terminal status, and when the A becomes idle, the application initiates a third party call to A and plays a message.

$3rdPartyCall_A \sqsubseteq \neg inArea_A$ – if the application initiates a third party call to A, then A is in the forbidden area.

A possible service interaction occurs when the user A is subscribed to PMF, and has forwarded her calls to user B. Initially both A and B are in the permitted area. When the user A enters the forbidden area, the PMF application is notified. Then the PMF application places a third party call to A, the call is forwarded to B which is in the permitted area.

VII. DETECTION OF WEB SERVICE INTERACTION

A knowledge representation system based on description logics is able to perform specific kinds of reasoning. For example, it is important to find out whether a newly defined concept makes sense or it is contradictory. From a logical point of view, a concept makes sense for us if there is some interpretation that satisfies the axioms of T (that is, a model of T) such that the concept denotes a nonempty set in that interpretation. A concept with this property is said to be satisfiable with respect to T and unsatisfiable otherwise.

Description logics as that for representation of mobile communication system behavior with negation and disjunction can be handled by so-called tableau-based algorithms. Instead of directly testing subsumption of concept descriptions, tableau-based algorithms use negation to reduce subsumption to (un)satisfiability of

concept descriptions: C ⊑D if ¬C ⊔D is unsatisfiable. A tableau method, shown in Table I, is used to detect feature interaction.

The tableau $t \stackrel{def}{=} \{ \langle b \mid p: C \rangle \}$ is a set of prefixed formulae where the prefix of given formula is consisted of a binary string $b := \varepsilon \mid (1|0)^+$ and a string of alternating names $p := n(Rm)^+$, and C is concept. Here ε is the empty string, n and m are names of individuals, R stands for the names of roles, and $()^+$ denotes one or more occurrences. Strict or relaxed prefix σ_1 of given string σ_2 can be defined by total ($\sigma_1 \prec \sigma_2$) or partial ($\sigma_1 \preccurlyeq \sigma_2$) order. Then b_M is called maximal for b in t if $b \in t \wedge b_M \in t \wedge b_M \prec b \wedge (\neg \exists b_1 \in t: b_M \prec b_1 \wedge b_1 \prec b)$.

The interaction between CFU and PMF services can be detected applying the following algorithm:

TABLE I
TABLEAU METHOD

AND:	$\dfrac{\langle b \mid p : C \sqcap D \rangle}{\langle b \mid p : C \rangle \quad \langle b \mid p : D \rangle}$	
OR:	$\dfrac{\langle b \mid p : C \sqcup D \rangle}{\langle b_M 0 \mid p : C \rangle \quad \langle b_M 1 \mid p : D \rangle}$	b_M maximal for b
SOME:	$\dfrac{\langle b \mid p : \exists R.C \rangle}{\langle b \mid pRn : C \rangle}$	pRn new (unless pR exists in the branch)
ALL:	$\dfrac{\langle b \mid p : \forall R.C \rangle}{\langle b \mid pRn : C \rangle}$	pRn present in b
KB:	$\dfrac{\vdots}{\langle b \mid p : \neg C \sqcup D \rangle}$	with p present in b and $C \sqsubseteq D \in T$

1. *Initial state*: Applying AND to the start formula $\langle \varepsilon \mid s_0: \sqcap_{A \in USERS} unreachable_A \rangle$ gives

   1.1 $\langle \varepsilon \mid s_0: unreachable_A \sqcap unreachable_B \rangle$

2. *User A attaches to the network:* To rule: $unreachable_A \sqsubseteq \exists statusNotificationReachable_A.idle_A$ applying KB gives

   $< \varepsilon \mid s_0: \neg unreachable_A \sqcup \exists statusNotificationReachable_A.idle_A >$ and applying OR:

   2.1 $<0 \mid s_0: \neg unreachable_A >$ which is closed.

   2.2 $<1 \mid s_0: \exists statusNotificationReachable.idle_A >$

   2.3 Applying SOME $<1 \mid s_0: statusNotificationReachable_A\ s_1: idle_A >$ with new state s_1

3. *User B attaches to the network:* To rule: $unreachable_B \sqsubseteq \exists statusNotificationReachable_B.idle_B$ applying KB gives

   $< \varepsilon \mid s_0: \neg unreachable_B \sqcup \exists statusNotificationReachable_B.idle_B >$ and applying OR:

   3.1 $<10 \mid s_0: statusNotificationReachable_A\ s_1: idle_A \sqcap (\neg unreachable_B) >$ which is closed.

   3.2 $<11 \mid s_0: : statusNotificationReachable_A\ s_1: idle_A \sqcap \exists statusNotificationReachable_B.idle_B >$

   3.3 Applying SOME $<1 \mid s_0: statusNotificationReachable_A\ s_1: statusNotificationReachable_B\ s_2: idle_A \sqcap idle_B >$ with new state s_2

4. *User B changes the location entering into permitted area:* To rule $idle_B \sqsubseteq \exists locationNotification_B.inArea_B$ applying KB gives

   $< 11 \mid s_0: statusNotificationReachable_A\ s_1\ statusNotificationReachable_B\ s_2: (\neg idle_B \sqcup \exists locationNotification_B.(idle_B \sqcap inArea_B) \sqcap idle_A) >$ and applying OR:

   4.1 $< 110 \mid s_0: statusNotificationReachable_A\ s_1\ statusNotificationReachable_B\ s_2: (\neg idle_B) \sqcap idle_A) >$ which is closed.

   4.2 $<111 \mid s_0: statusNotificationReachable_A\ s_1\ statusNotificationReachable_B\ s_2: (\exists locationNotification_B.(inArea_B \sqcap idle_B)) \sqcap idle_A$

   4.3 Applying SOME $<111 \mid s_0: statusNotificationReachable_A\ s_1\ statusNotificationReachable_B\ s_2\ locationNotification_B\ s_3\ (inArea_B \sqcap idle_B \sqcap idle_A) >$ with new state s_3

5. *User A changes the location entering into forbidden area:* To rule

   $idle_A \sqsubseteq \exists locationNotification_A.(\neg inArea_A \sqcap idle_A)$ applying KB gives

   $< 111 \mid s_0$: $statusNotificationReachable_A\ s_1\ statusNotificationReachable_B\ s_2$: $locationNotification_B\ s_3$

   $(\neg idle_A \sqcup \exists locationNotification_A.(idle_A \sqcap \neg inArea_B) \sqcap (idle_B \sqcap inArea_B)>$ and applying OR:

   5.1 $< 1110 \mid s_0$: $statusNotificationReachable_A\ s_1\ statusNotificationReachable_B\ s_2$:

   $(\neg idle_A) \sqcap idle_B \sqcap inArea_B)>$ which is closed.

   5.2 $<1111 \mid s_0$: $statusNotificationReachable_A\ s_1\ statusNotificationReachable_B\ s_2$:

   $(\exists locationNotification_A.(\neg inArea_A \sqcap idle_A)) \sqcap (idle_B \sqcap inArea_B)$

   5.3 Applying SOME $<1111 \mid s_0$: $statusNotificationReachable_A\ s_1\ statusNotificationReachable_B\ s_2$

   $locationNotification_B\ s_3\ locationNotification_B\ s_3$: $inArea_B \sqcap idle_B \sqcap idle_A(\neg inArea_A)>$ with new state s_4

6. *A third party call is initiated to A and a message is played to B:* To rule

   $idle_A \sqcap (\neg inArea_A)] \sqsubseteq \exists makeCallSession_A$. $3rdPartyCall_A \sqcap \exists playAudioMessage_A.3rdPartyCall_A$ Applying KB, OR, SOME twice and CFU_{AB} it is derived:

   $<111111 \mid s_0$: $statusNotificationReachable_A\ s_1\ statusNotificationReachable_B\ s_2$

   $locationNotification_B\ s_3\ locationNotification_B\ s_3\ makeCallSession_A\ s_4\ 3rdPartyCall_B \sqcap inArea_B$

   $playAudioMessage_B\ s_5$: $3rdPartyCall_B \sqcap inArea_B >$ which is closed because of

   $3rdPartyCall_B \sqsubseteq \neg inArea_B$.

The result is a closed tableau which means that refinements of CFU and PMF in the context of Parlay X Web Services "Third party call", "Audio call" and "Terminal location" interact on activation $\{CFU_{AB}\} \cup \{PMF_A\}$. It is important to mention that the service interaction can be detected automatically since the programmability of the algorithm.

Having statements in the Tbox and web service functionality both described as policy rules, Parlay X "Policy" interfaces may be used to evaluate policy.

A third party application defines policy representing the web service behavior in the network by the use of PolicyProvisioning interface.

The PolicyEvaluation interface may be used by the application during specification time to evaluate the behavior of web services and to detect possible web service interaction before deployment if the network. The evaluatePolicy operation is used to request an evaluation of a rule, as shown in Fig.2.

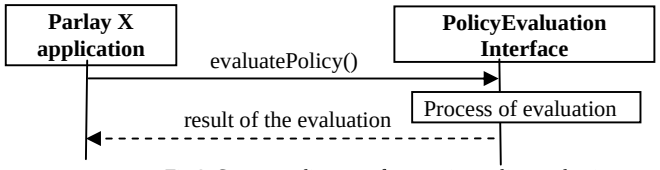

Fig.2. Sequence diagram of requesting policy evaluation

During execution time a third party application makes subscription to policy related events. The PolicyEventNotificationManager interface is used for starting and ending subscription of the notification about events, as shown in Fig.3. The application is notified about policy related events using the operation of PolicyEventNotification interface. Notified about occurred web service interaction in the network, the application may invoke specific actions to resolve the problem.

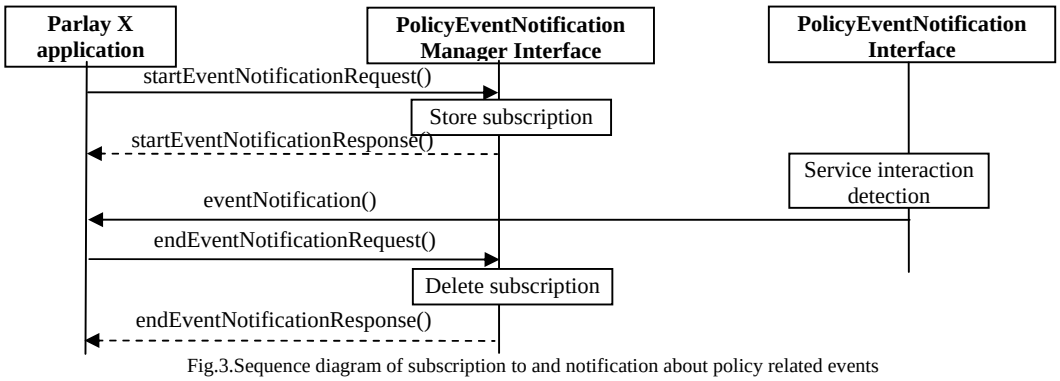

Fig.3.Sequence diagram of subscription to and notification about policy related events

VIII. CONCLUSIONS

Web service interaction problem may be managed by policies. Policies are used to describe web service behavior. Web Service interaction is detected by policy evaluation. Parlay X "Policy" interfaces may be used for this purpose. The suggested approach provides a formal method for detection of interactions in the context of call and user interaction control and mobility management. Policy rules describe the application view on the call and session states, as wells on the mobility management states. Description logic provides a formal tool for representing the behaviour in different communication scenarios. The Parlay X application interaction detection is presented as standard reasoning task. Future work will be aimed at formal specification of Parlay X web services related to quality of service control, messaging, presence and availability management which will expand the range of Parlay X applications that may be explored.

REFERENCES

[1] S. Apel, C. Kästner, "An overview of feature-oriented software development", *Journal of Object Technology*, vol. 8 issue 5, pp. 49–84, 2009.

[2] S. Apel, W. Scholz, C. Lengauer, C. Kästner. "Detecting dependences and interactions in feature-oriented design", in *Proc. International Symposium on Software Reliability Engineering ISSRE'2010*, pp. 161–170, 2010.

[3] S., Apel, A. Rhein, T. Thum, C. Karstner, "Feature-interaction detection based on feature-based specification", *Journal on Computer Networks*, Elsevier, vol.57 pp.2399-2409, 2013.

[4] S. Apel, H. Speidel, P. Wendler, A. von Rhein, D. Beyer, "Detection of feature interactions using feature-aware verification", in *Proc. of International Conference on Automated Software Engineering ASE'2011*, pp. 372–375, 2011.

[5] Z. Chentouf, "Detecting OAM&P design defects using a feature interaction approach", *Journal of Network Management*, vol. 22, issue 2, pp.95-103, 2012.

[6] R. Glushko, K. Nomorosa, "Substituting Information for Interaction: A Framework for Personalization in Service Encounters and Service Systems", *Journal of Service Research*, vol.16, no. 1 pp. 21-38, 2013.

[7] W. Scholz, T. Thüm, S. Apel, C. Lengauer, "Automatic detection of feature interactions using the java modeling language: an experience report", in *Proc. of International Workshop on Feature-Oriented Software Development FOSD'2011*, pp.7:1–7:8, 2011.

[8] N. Siegmund, S, Kolesnikov, C. Kästner, S. Apel, D. Batory, M. Rosenmüller, G. Saake, "Predicting performance via automated feature-interaction detection", in *Proc. of International Conference on Software Engineering, ICSE'2012*, pp. 167-177, 2012.

[9] F. Takeyama, S. Chiba, "Implementing Feature Interactions with Generic Feature Modules", *Lecture Notes on Computer Science*, Springer, 8088, pp.91-96, 2013.

[10] Aalst, W., A. Mooij, C. Stahl, K. Wolf, "Service Interaction: Patterns, Formalization, and Analysis", Retrieved from: http://www.win.tue.nl/~cstahl/Papers/AalstMSW-sfm2009.pdf, 2009.

[11] Z. Leila, J. Davis, "Modeling Service Interaction Networks", in *Proc. of Americas Conference on Information Systems AMCIS'2012*, Retrieved from: http://aisel.aisnet.org/amcis2012/proceedings/ VirtualCommunities/8, 2012.

[12] J. Hu, W. Yu, C. Kun, S. Reiff-Marganiec, "Web Services Feature Interaction Detection based on Situation Calculus", in *Proc. World Congress on Services (SERVICES-1)*, pp.213-220, 2010.

[13] 3GPP TS 23.218 IP Multimedia (IM) session handling; IM call model, v9.1.0, Release 9, 2009.

[14] 3GPP TS 29.199-22 Open Service Access (OSA); Parlay X Web Services, Part 22: Policy, Release 9, 2009.

[15] 3GPP TS 29.199-2 Open Service Access (OSA); Parlay X Web Services, Part 2: Third party call, Release 9, 2009.

[16] 3GPP TS 29.199-11 Open Service Access (OSA); Parlay X Web Services, Part 11: Audio call, Release 9, 2009.

[17] 3GPP TS 29.199-8 Open Service Access (OSA); Parlay X Web Services, Part 8: Terminal status, Release 9, 2009.

[18] 3GPP TS 29.199-9. (2009). Open Service Access (OSA); Parlay X Web Services, Part 9: Terminal location, Release 9, 2009.

