Checking Compatibility and Replaceability in Web Services Business Protocols with Access Control*

Emad Elabd, Emmanuel Coquery, Mohand-Said Hacid

University Claude Bernard Lyon 1, LIRIS
43, bd du 11 novembre 1918
69622 Villeurbanne cedex, France
eelabd,emmanuel.coquery,mshacid@liris.cnrs.fr

Abstract. Recently, describing behavior of web services is becoming more and more important. This behavior can be described by business protocols representing the possible sequences of message exchanges. Since a lot of web services use access control policies to restrict the access to authorized consumers, these policies should be part of the service description. Studying the behavior of web services by analyzing their business protocol after assigning the access control policies is the main contribution of this work. Access control policies will be presented using ontology which eases policy specification and management and add some flexibility in the policy comparison. This paper introduces notions of compatibility and replaceability w.r.t. business protocols with access control policies annotations, together with the corresponding verification algorithms.

1 Introduction

1.1 Web services

Web services are loosely coupled applications that use XML based technology for representation and communication across the Internet. Web services technology is emerging as main pillar of service-oriented architectures (SOA) [3]. Services in SOA need richer description models due to the loose coupling property of SOA. All the information about the service and needed by the client should be included in service descriptions in order to allow the client to interact correctly with the service. The business protocol of the web service that is the possible message exchange sequences supported by the service should be included in the service descriptions such as proposed in [3].

The semantic web is not a separate web but an extension of the current one, in which information is given well-defined meaning, better enabling computers and people to work in cooperation [4]. Several researchers have recognized that web services standards lack of semantics [5]. Semantic description allow better performance in automatic service discovery, composition, invocation and monitoring. Business applications whose functionality is semantically described can be found and integrated more easily than those without semantic descriptions. This presents opportunities for semantic web

* This work is partially founded by the french ANR project ServiceMosaic ANR-06-JCJC-0075-01 [1] and the european FP7 project Compas GAN 215175[2].

services in integrating enterprise systems. Therefore, augmenting the web service description with the access control policy(ACP) requirements is a contribution towards semantic web services. This work discusses the results of modeling and analyzing web service business protocols augmented with time constraints and access control policy. Access control policies are expressed using ontologies in order to benefit from the flexibility offered by subsumption on concepts together with the possibility to use ontology alignment in the context of the semantic Web. We define and verify web service compatibility in order to see if (and how) two services can have interactions based on their protocols. We also define and verify a notion of replaceability to see if a service can be replaced transparently by another one for its current requesters between business protocols after adding access control policy to their descriptions. Therefore, our main contribution is to model and analyze web services after assigning the ACP in the business protocols.

1.2 Web service business protocols modeling and analyzing

The need for formal methods and software tools for automatically analyzing service descriptions is widely recognized, and many approaches have been developed to this end. Formalisms allow us to reason with the constructed models, analyzing and verifying some properties of interest of the described systems. Timed automata [6] are well known formalisms for real-time systems and there are some well-known tools supporting them as UPPAAL [7]. Therefore, it can be used for describing and analyzing the behavior of web services, specifically those including time restrictions. Previoulsy, a model for business protocols and a framework for protocol-based analysis had been presented by Benatallah et al. [8,9,10] and Karim Baina et al. [11]. They studied the compatibility and replaceability issues. This model captures all of the conversations that is supported by a service. The model is presented using state chart which is a suitable model for describing behaviors and they used timed automata later. Figure 1 shows an example of two web service business protocol P1 and P2 [1]. States represent the various stages that a service may go through while transitions are triggered when a message is received or sent. There is a unique initial state(e.g. **start** in P1 and P2) and many final states(e.g. **orderpaid** and **ordercanceled** states in P1 and P2). In protocol P1 for example, conversation starts by sending **orderGoods(-)** message with polarity (-) which indicates that the message is a send message. After that, a **makePayment(-)** message is sent. On the other side, service P2 received the two sent messages. Every conversation ends in final state is said to be valid. According to the definition of compatibility defined by Benatallah et al. [8,9,10], the two protocols P1 and P2 in figure 1 are fully compatible(i.e all the executions of P1 can interoperate with P2). Also, in figure 2 the two protocols are partially compatible because there is at least one possible conversation can take place among a service supporting P1 and a service supporting P2. Figure 2 shows that protocol P2 has not the ability to receive the **cancelOrder(-)** message from P1. As a result, there is a potential conversation will not be accomplished and results an error. Therefore, in our work we will adapt a new definition of compatibility based on the error free interaction. In other words, two business protocols are compatible if and only if any potential send message from one protocol can be received by the other protocol during their interaction and vice versa. Based on this definition, the two protocol of

figure 2 are not compatible because, during the interaction, when the first protocol P1 is on the state **(ordersend)** and P2 is in the state **(ordermade)** the message **(cancelOrder)** may be sent and in this case an error will happen because P2 can not receive the message. Julien Ponge et al. in [12]extended the work done by Benatallah et al. [8,9,10] by presenting a formalization of the protocols that include time-related constraints , and the impact of time on compatibility and replaceability analysis. They formalized the C-Invoke constraints which define time windows of availability and M-Invoke constraints which define expirations deadlines.

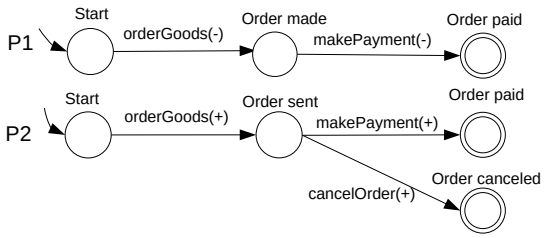

Fig. 1. Two compatible web services without problem (full compatibility).

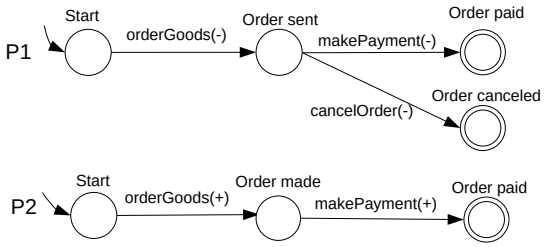

Fig. 2. Two web services incompatible.

This paper is organized as follows. Section 2 shows an informal scenario to explain the importance of assigning the access control policy in business protocol. Section 3 lists some access control models proposed in a previous work for securing web service and then explains how to use the description logics as an ontology in presenting access control policy and lists the benefits of using ontology in policy specifications and management. Section 4 describes the formal methods and algorithms which are used in analyzing and modeling the web service business protocol. Finally, section 5 presents the conclusion.

2 Informal Scenario

This section presents an informal simple scenario of the interaction between a service and a consumer. With this example, we will illustrate the importance of assigning the ACP in the web service protocol. The web service that is used in this example is for accessing university library papers. The business logic of the service consists in logging as student or professor then you can access journal and conference papers if you are professor and access conference papers if you are student. The access control policy for this example is presented in figure 3. Figure 4 presents the business protocols of the service (P1) and the consumer (P2) without assigning the access control policy. Based on the definition of compatibility, these two protocols are compatible because any potential message will be sent by the consumer can be received by the provider and vice versa. This type of analysis does not consider the ACP in checking the compatibility. Therefore, this definition of compatibility between two protocols will not guarantee that the consumer can access the required resource because the business protocol does not describe the ACP for accessing the resources.

Figure 5 presented the service business protocol (P1) and the consumer business protocol (P2) after assigning the ACP. As shown, if we do not consider the ACP in checking the compatibility between the service and the consumer, we will say that they are compatible but if we consider the ACP, we will say that they are incompatible because the required credential in the serviced as in its protocol P1 is **prof** credential and the provided credential by the consumer in P2 is a **student** credential. As a result, a new definition of the compatibility taking into account the access control policy is needed to guarantee the access of the required resources with the provided credential during the analysis phase. Assigning access control policy is also important in replaceability analysis. We present two main scenarios to indicate the importance of assigning the access control policy in replaceability analysis:

- Service provider wants to replace his old service with new one supports the same conversations with the same ACP (e.g. the new service has more functionality than the old one but all the functionality of the old one must included in the new one).
- Consumer wants to replace web service at which he interacts with a new one supports the same conversations with the same ACP (e.g. the new service has more quality of service more simple interface).

So, by using this type of business protocol presentation, we can check compatibility and replaceability in terms of message exchanged and Access control policy.

Access Control Policy:
- For login: professor credential or student card.
-For accessing journal papers: professor credential
-For access conference papers: professor credential or student card

Fig. 3. An access control policy for accessing library web service.

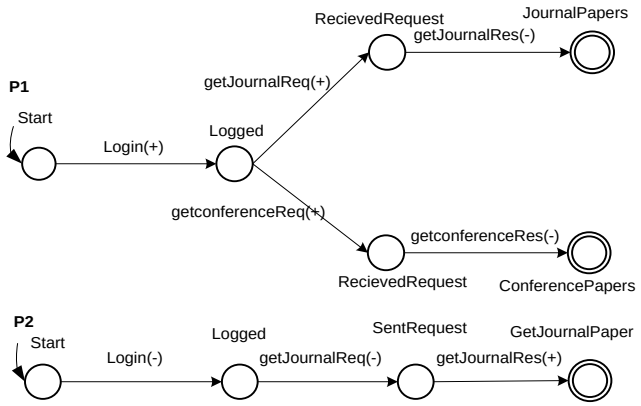

Fig. 4. A web service business protocol (P1) and a consumer business protocol (P2) without assigning the ACP.

3 Access Control Policy

Development of suitable access control models, able to restrict access to Web services to authorized users is an important issue. Security technologies commonly adopted for Web sites and traditional access control models are not satisfactory [13]. Currently, There are two research directions in access control. One has focused on efforts to develop new access control models to meet the policy needs of real world application domains. These have led to several successful models such as the NIST Standard RBAC model [14], the RBAC96 model [15], WS-AC1 [13], and the RT model [16]. In a parallel, researchers have developed policy languages for access control. These include XACML [17], Ponder [18], WS-Policy [19] and finally to Semantic Web based languages such as Rei [20], DARPA agent markup language for services (DAML-S),and KAoS [21]. Policy languages grounded in Semantic Web technologies allow policies to be described over heterogeneous domain data and promote common understanding among participants who might not use the same information model. There are two main advantages of using ontology in policy specification and management [22]:

1. Ease policy specification and management by sharing policies for common attributes, composing and overriding policies.
2. Protect sensitive information by avoiding information leaking request and answering unnecessary request.

Description logics (DLs) [23] as policy formalisms technique can be used to present access control policy ontology. Descriptions logics are very useful for defining, integrating, and maintaining ontology, which provide the Semantic Web with a common understanding of the basic semantic concepts used to annotate Web pages. Access control policy will be formalized using the DL. Knowledge representation system based on DLs consists of two components - TBox and ABox. The TBox describes terminology,

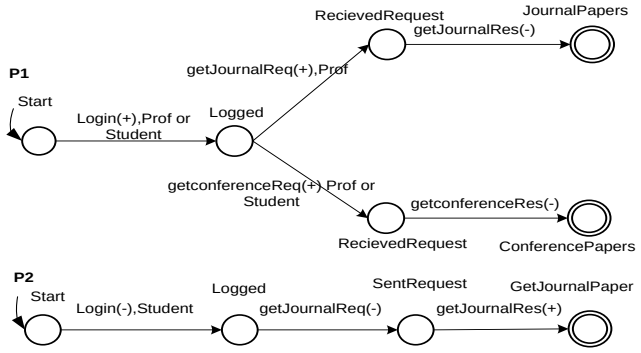

Fig. 5. A web service business protocol (P1) and a consumer business protocol (P2) after assigning the ACP.

i.e., the ontology in the form of concepts and roles definitions, while the ABox contains assertions about individuals using the terms from the ontology. Concepts describe sets of individuals, roles describe relations between individuals. The access control policies are presented as concepts describe set of credentials as individuals. we will perform subsumption($\sqsubseteq$), union($\sqcup$), and intersection($\sqcap$) operations on the ontology during our algorithms. In the next sections, an example of checking compatibility with the aid of ontology is presented. Presenting ACP of web services as ontology will enables us to use ontology alignment tools to find classes of data that are "semantically equivalent ". We can use the ontology of the service provider and the ontology of the consumer and produce new global ontology. This new ontology can be used on our analysis in checking the compatibility between web service and the consumer.

4 Formalization and Algorithms

This section presents the formal definitions of timed business protocol and the algorithms used in the automated analysis. The Timed business protocol definitions is based on the definition of Benatallah in [3], augmented with ACP. The timed business protocol is represented as a state chart which consists of a set of states containing an initial state and one of more of final states and set of transitions. States represent the various stages that a service may go through while transitions can be implicit transition (i.e. an internal transition of the service from one state to another without sending or receiving messages) or explicit transition which are triggered when a message is received or sent. The implicit transitions are assigned with time constraints and the explicit transition is assigned with message specifications and the ACP. This protocol is deterministic (i.e. all the outputs transition from any state are different) and does not contain any cycle constituted with only implicit transitions.

Definition 1. *Timed business protocol (TBP) assigned with ACP is a tuple* $Pr = (S; s_0; T; F)$ *which consists of the following elements:*

- S *is a finite set of states.*
- $s_0 \in S$ *is the initial state.*
- $T = T_e \cup T_i$ *, where T is a finite set of transitions with:*
  - $T_e \subseteq S^2 \times M \times (\{-\} \times 2^c) \cup (\{+\} \times P))$ *, a finite set of explicit transitions where M is a set of messages, assigned to the explicit transitions between the states, P is the set of access control policies, C is the set of credentials. If there is no policy or credential it will be assigned the default value* $\emptyset$ *for negative transitions and* $\top$ *for positive transitions.*
  - $T_i \subseteq S^2 \times N : t$*, a finite set of implicit transitions (i.e. an internal transition of the service from one state to another without sending or receiving messages) , where* N *is the set of implicit transitions names assigned to the implicit transitions between the states and* $t \in Q$ *is the time .*
  - *This protocol is deterministic (i.e. all the outputs transition from any state are different) and does not contain any cycle constituted with only implicit transitions.*
  - *All states in the automata are accessible and co-accessible.*
- $F \subseteq S$ *is a set of final states. If* $F = \{\phi\}$ *then* Pr *is said to be an empty protocol.*

The conversion can be achieved by applying algorithm 1. In order to ease the analysis of such protocols, we perform the conversion of implicit transitions to time constraints on explicit transitions. The new new business protocol is called "Explicitly Timed Business Protocol assigned with ACP. The conversion algorithms is consists of two main steps:

1. Updating all the explicit transitions which share with an implicit transition the same source state by adding time constraints reflects the effect of the implicit transitions.
2. Update the time constraints on the explicit transitions which have a preceding implicit transitions or implicit path (i.e. there is an implicit transition or path before the source state of an explicit transition).

Complexity analysis: Let n be the number of states and the numbers of the implicit transitions and explicit transition are L_e and L_i respectively. Therefore, the conversion algorithm runs in time $O(L_e * L_i)$.

Definition 2. *An explicitly time business protocol assigned with ACP is a tuple* $Pr = (S; s_0; T; F)$ *which consists of the following elements:*

- S *is a finite set of states.*
- $s_0 \in S$ *, is the initial state.*
- $T \subseteq S^2 \times M \times (\{-\} \times 2^c) \cup (\{+\} \times P)) \times I$ *, is a finite set of explicit transition where M is a set of messages assigned to the explicit transitions between the states, P is the set of access control policies, C is the set of credentials. I is the set of time intervals in the form* $I[x, y[$ *where* $x, y \in \Re^+ \cup +\{\infty\}$.
- *This protocol is deterministic (i.e. all the outputs transition from any state are different) and does no contain any cycle constituted with only implicit transitions.*
- *All states in the automata are accessible and co-accessible.*
- $F \subseteq S$ *is a set of final states. If* $F = \{\phi\}$ *then* Pr *is said to be an empty protocol.*

$output(s_i)$ defines all the outgoing transitions triggered from the state (s_i) and $input(s_i)$ defines all the incoming transitions to the state (s_i).

Figure 6 and figure 7 show an example of a business protocol with implicit transition and its equivalent business protocol without implicit transition respectively.

Algorithm 1: Conversion of timed business protocol Pr with implicit transition to explicitly business protocol.

//Updating all the explicit transitions which share with an implicit transition the same source state for each state $S_i \in S$

foreach *state* $s_i \in S$ **do**

  if $\exists(s_i, s_j, t) \in T_i$ *where* $s_j \in S$ **then**

    forall $(s_i, s_k, m_k^{\pm}, I_k, pc_k) \in T_e$ *where* $s_k \in S$, $m_k \in M$,$0 \leq k \leq n$, n *number of states in pr,*pc_k *is the ACP and* $m^{\pm}$ *means that the message either output or input* **do**

      $I_k = [0, t[$

  else

    forall $(s_i, s_k, \overleftrightarrow{m}_k, I_k, pc_k) \in T_e$ *where* $s_k \in S$, $m_k \in M$,$0 \leq k \leq n$,*and* n *number of states in pr* **do**

      $I_k = [0, \infty[$

//Update explicit transitions which have preceding implicit transitions or paths(i.e. there is an implicit transition or path before the source state of an explicit transition).

while $\exists IT(s, s', t) s.t \nexists IT(s", s, t')$ **do**

  foreach $(s', s", \overleftrightarrow{m}, I(x, y), pc) \in T_e$ **do**

    $T_e = T_e \cup (s', s", \overleftrightarrow{m}, I(x + t, y + t), pc)$

  Delete transition IT(s,s',t)

Return Pr

4.1 Compatibility

There is a difference in the methodology between checking the compatibility in terms of message exchange and in terms of ACP. Checking the compatibility in terms of message exchange depends on the current message and corresponding current message. But checking the compatibility in terms of ACP depends on the current ACP and the previous or current credentials of the corresponding transition. Therefore, there is a need to update each transition with all the credentials that can be provided before reaching it (i.e. transition credentials updated to be all the credential resulted from the current credentials and the previous provided credentials). We called this set of credential **cumulative ACP**. Figure 8 shows an example of two business protocols P1 and P2 where P1 provide its X credential in the first transition but P2 asked for it in the third transition. If we compare the two protocols without calculating the cumulative ACP then we will find that they are incompatible but after calculating the ACP for P1 we find that they are compatible. In figure 9, P1 is a client protocol and P2 is a service protocol and P1 is compatible with p2. $P^{'}2$ is another service protocol which is not compatible with P1. The aim of this example is to indicate that when comparing ACP and credential the ACP must satisfy the credentials. The problem is that the cumulative ACP may give us an ACP on transition and the corresponding credential will not satisfy it but the two protocols are compatible. This is shown in figure 10 where the two protocols are compatible. The reason for this problem is that we calculate the cumulative before determining the transition which will be used in the interaction between the two protocols. So, when we

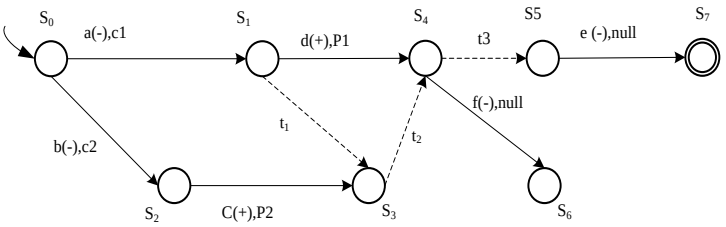

Fig. 6. Business protocol of web service with implicitly transition.

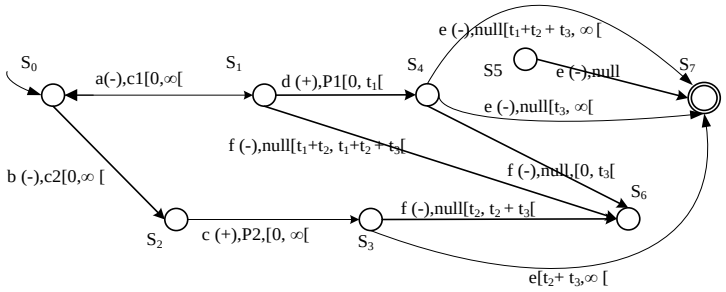

Fig. 7. Business protocol of web service without implicitly transition.

update the cumulative ACP, we only consider the transition which will potentially share in the interaction.

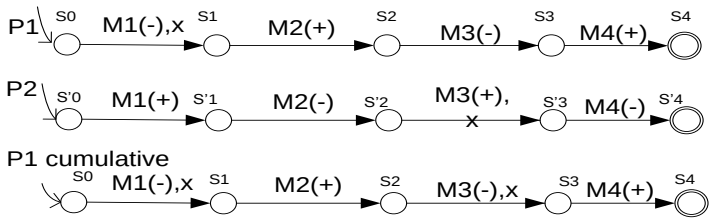

Fig. 8. Two protocols indicate the importance of calculating the cumulative access control.

Now, we introduce some formal definitions leading to two definitions of compatibility. The first one, definition 4, expresses compatibility in terms of the guarantees that it provides: it ensures that messages that can be sent can also be received and that there is no dead or live lock. The second one, definition 8, is used as a base for the algorithm that checks compatibility.

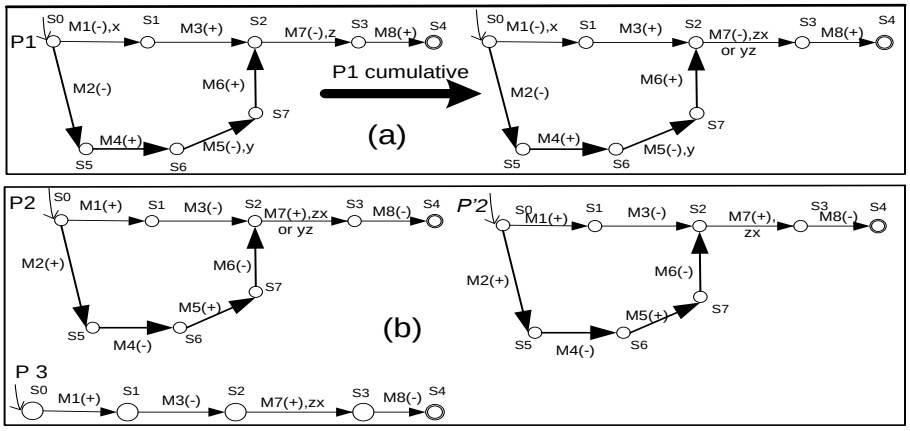

Fig. 9. (a)P1 before and after calculating the cumulative ACP (b) Three different protocols P2 , $P^{'}2$, and P3. P1 is compatible with P2 but not compatible with $P^{'}2$. P3 is not compatible with P1 because it cannot receive m2. Therefore P3 is not compatible in terms of message and in ACP.

Definition 3. *An interaction trace IT between a protocol $P^1 = (S^1, s_0^1, T^1, F^1)$ and a protocol $P^2 = (S^2, s_0^2, T^2, F^2)$ is a finite sequence $((s_i^1, s_i^2, \overleftrightarrow{m_i}, c_i, t_i, s_{i+1}^1, s_{i+1}^2))_i$, where $s_n^1 \in S^1$ and $s_n^2 \in S^2$. m_i is a message instance with its direction (either $\leftarrow$ or $\rightarrow$). c_i is the set of all credentials sent either in this message or in any previous message with the same direction. t_i is the time period since the previous message, or 0 for the first message.*

Definition 4. *An interaction trace $IT = ((s_i^1, s_i^2, \overleftrightarrow{m_i}, c, t, s_{i+1}^1, s_{i+1}^2))_i$ between a protocol $P^1 = (S^1, s_0^1, T^1, F^1)$ and a protocol $P^2 = (S^2, s_0^2, T^2, F^2)$ is* correct *if and only if for every tuple $(s^1, s^2, \overrightarrow{m}, c, t, s'^1, s'^2)$ in IT (and symmetrically for every tuple $(s^1, s^1, \overleftarrow{m}, c, t, s'^1, s'^2)$):*

- *there are two transitions $(s^1, s'^1, \mathrm{M}^-, \mathrm{C}, \mathrm{I}^1) \in T^1$ and $(s^2, s'^2, \mathrm{M}^+, \mathrm{P}, \mathrm{I}^2) \in T^2$;*
- *m is an instance of* M*;*
- *the credentials sent in m match* C *and c is a instance of* P*;*
- $t \in \mathrm{I}^1 \cap \mathrm{I}^2$.

The set of correct interactions traces between P^1 and P^2 is noted $IT(P^1, P^2)$.

IT is said to be complete *if for its last tuple $(s_{n-1}^1, s_{n-1}^1, \overrightarrow{m_{n-1}}, c_{n-1}, t_{n-1}, s_n^1, s_n^2)$, $s_n^1 \in F^1$ and $s_n^2 \in F^2$*

Definition 5. *(Compatibility in Terms of Interaction Trace Assigned with ACP.) Two business protocols, $P^1 = (S^1, s_0^1, T^1, F^1)$ and $P^2 = (S^2, s_0^2, T^2, F^2)$ are compatibles in terms of interaction trace if $\forall tr \in IT(P^1, P^2)$, its last tuple $(s_{n-1}^1, s_{n-1}^1, \overrightarrow{m_{n-1}}, c_{n-1}, t_{n-1}, s_n^1, s_n^2)$ verifies:*

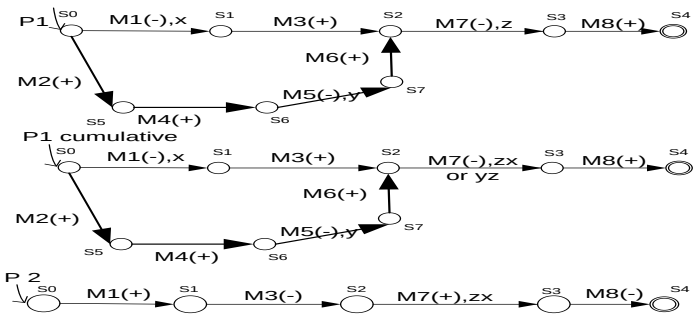

Fig. 10. P1 is compatible with P2 (this show the importance of calculating the cumulative ACP after determining the transition which will be used in the interaction and this is accomplished by product automata.)

- $\forall m \forall t$ *if* $\exists(s_n^1, s^1, \mathrm{M}^-, \mathrm{C}^1, \mathrm{I}^1) \in T^1$ *such that the set of credentials* c *sent in* m *matches* C^1, $t \in \mathrm{I}^1$ *and* m *is an instance of* M *then* $\exists s^2 \in S^2$ *such that* $tr.(s_n^1, s_n^2, \overrightarrow{m}, c, t, s^1, s^2) \in IT(P^1, P^2)$
- $\forall m \forall t$ *if* $\exists(s_n^2, s^2, \mathrm{M}^-, \mathrm{C}^2, \mathrm{I}^2) \in T^2$ *such that the set of credentials* c *sent in* m *matches* C^2, $t \in \mathrm{I}^2$ *and* m *is an instance of* M *then* $\exists s^1 \in S^1$ *such that* $tr.(s_n^1, s_n^2, \overleftarrow{m}, c, t, s^1, s^2) \in IT(P^1, P^2)$

Definition 6. *The* ***product automata*** A^p *of two timed business protocols* $P^1 = (S^1, s_0^1, T^1, F^1)$ *and* $P^2 = (S^2, s_0^2, T^2, F^2)$ *is defined as* $A^p = (S^p, s_o^p, T^p, F^p)$ *where:*

- $S^p = S^1 \times S^2$
- $s_o^p = (s_0^1, s_0^2)$
- T^p *is the greatest subset of* $((S^1 \times S^2) \times (S^1 \times S^2) \times ((\overrightarrow{M} \times 2^{c^1} \times p^2) \cup (\overleftarrow{M} \times 2^{c^2} \times p^1)) \times I)$ *such that for for all transition* $((s_i^1, s_i^2), (s_{i+1}^1, s_{i+1}^2)), \overleftrightarrow{m}_i, p^p, c^p, I^p) \in T^p$ *there exist two transitions* $(s_i^1, s_{i+1}^1, m_i, (p^1 \text{ or } c^1), I^1) \in T^1$ *and* $(s_i^2, s_{i+1}^2, m_i, p^2 \text{ or } c^2)), I^2) \in T^2$ *with :*
  - $I^p = I^1 \cap I^2$
  - *polarity*$(m_i, P^1) \neq polarity(m_i, P^2)$ *and*
    - *If polarity* $(m_i, P^1) = -$ *then* $\overleftrightarrow{m}_i = \overrightarrow{m}_i$, $p_i^p = p_i^2$ *and* $c_i^p = c_i^1$
    - *otherwise* $(m_i, P^2) = -$, $\overleftrightarrow{m}_i = \overleftarrow{m}_i$, $p^p = p_i^1$ *and* $c_i^p = c_i^2$
- $F^p = F^1 \times F^2$

Definition 7. *(Cumulative path in the product automata):* $PA^p = ((s_i^1, s_i^2) \overset{(C_i^1, C_i^2)}{\rightarrow} (s_{i+1}^1, s_{i+1}^2), ..., (s_n^1, s_n^2) \overset{(C_n^1, C_n^2)}{\rightarrow} (s_{n+1}^1, s_{n+1}^2)))$ *is a cumulative path in the product automata* $A^p = (s^p, s_o^p, T^p, F^p)$ *where*

- *States* $(s_i^1, s_i^2), ..., (s_n^1, s_n^2) \in S^p$
- *Each credential* C_i^1 *is the set of cumulative credentials which is the union of the previous set of cumulative credentials* C_{i-1}^1 *and the current set of credentials* c_i^1

where c_i^1 is the set of credentials on the transition between the state (s_i^1 and s_{i+1}^1) of the protocol p^1 and $C_0^1 = c_0^1$ and each credential C_i^2 is the set of cumulative credentials which is the union of the previous set of cumulative credentials C_{i-1}^2 and the current set of credentials c_i^2 where c_i^2 is the set of credentials on the transition between the state (s_i^2 and s_{i+1}^2) of the protocol p^2 and $C_0^2 = c_0^2$.

- *A complete cumulative path in the product automata is the cumulative path which starts with the initial state (s_0^1, s_0^2) and ends with a final state $(s_f^1, s_f^2) \in F^p$.*

Definition 8. *(Co-accessibility of a State in the Product of Automata): $A^p = P^1 \times P^2 = (s^p, s_o^p, T^p, F^p)$ is the product of automata of two TBP P^1 and P^2, state $(s_i^1, s_i^2) \in S^p$ is co-accessible if there exist two paths PA^1 and PA^2 where $PA^2 = ((s_i^1, s_i^2).PA^1.(s_f^1, s_f^2))$ and $(s_f^1, s_f^2) \in F^p$.*

Definition 9. *(Compatibility in Terms of Product Automata assigned with ACP) Protocols $P^1 = (S^1, s_0^1, T^1, F^1)$ and $P^2 = (S^2, s_0^2, T^2, F^2)$ are two Timed business protocol, and $A^p = P^1 \times P^2 = (s^p, s_o^p, T^p, F^p)$ is a product automata assigned with ACP, we say that P^1 and P^2 are compatible using their product automata if there is a relation $R = S^1 \times S^2$ where for all $(s_i^1, s_i^2) \in R$:*

- *$\forall(s_i^1, s_{i+1}^1, m_i^-, c_i^1, I_i^1) \in T^1 \exists(s_i^2, s_{i+1}^2, m_i^+, p_i^2, I_i^2) \in T^2$ where $I_i^1 \subseteq I_i^2$ and $(s_i^1, s_i^2, s_{i+1}^1, s_{i+1}^2, \overrightarrow{m}_i, c_i^1$ and $p_i^2, I_i^1) \in T^p$ and $(s_{i+1}^1, s_{i+1}^2) \in R$.*
- *$\forall(s_i^2, s_{i+1}^2, m_i^-, c_i^2, I_i^2) \in T^2 \exists(s_i^1, s_{i+1}^1, m_i^+, p_i^1, I_i^1) \in T^1$ where $I_i^2 \subseteq I_i^1$ and $(s_i^1, s_i^2, s_{i+1}^1, s_{i+1}^2, \overleftarrow{m}_i, c_i^2$ and $p_i^1, I_i^2) \in T^p$ and $(s_{i+1}^1, s_{i+1}^2) \in R$.*
- *$(s_{i+1}^1, s_{i+1}^2) \in s^P$ is co-accessible, there is a path in the product automata from this state to final state.*
- *$(s_0^1, s_0^2) \in R$.*
- *For all the complete cumulative paths $PA^p = ((s_0^1, s_0^2) \overset{C_0^1, C_0^2)}{\rightarrow} (s_1^1, s_1^2), ..., (s_n^1, s_n^2) \overset{(C_n^1, C_n^2)}{\rightarrow} (s_{n+1}^1, s_{n+1}^2))$ in the product automata, each policy p_i^1 is satisfied by the set of cumulative credentials C_i^2 and p_i^2 is satisfied by the set of cumulative credentials C_i^1.*

The algorithm which is used for checking the compatibility between two protocols in terms of product automata with ACP can be divided into two parts. The first part is for checking compatibility in terms of message exchange and this can be done by constructing the product automata and traversing through it, starting by the initial state, using breadth first approach and checking that if there is a state does not included in this relation set R (i.e each state have two corresponding states of the two protocol and all the outgoing messages from this state in one protocol can be received by the another protocol) then the algorithm stops and the two protocols are not compatible else if all states in the product automata are included in this relation set then the two protocols are compatible in terms of message exchange and goto the second part. The second part is for calculating the cumulative credentials on each transition on the product automata. The idea of this part is to use the queue data structure to cumulate the credentials. Each element of the queue consists of the state, cumulative credentials corresponding the protocol P^1 in this state, and the cumulative credential corresponding the protocol P^2in

this state. The algorithm traverses through the automata to for updating these credentials of the states and in the same time update the cumulative credentials on the transitions. After calculating the cumulative credentials on each transition, if any ACP related to one of the two protocols on any transitions is not satisfied by the cumulative credentials on this transition related to the other transition then the two protocol are not compatible in terms of ACP. Algorithm 2 presents the first part of the algorithm and Algorithm **??** presents the second part of the algorithm.

Complexity analysis: Let T1 and T2 be the number of transitions of the two protocols P^1 and P^2 respectively, the construction of the product automata will take (T1 $\times$ T1). The calculation of the cumulative credentials will take number of states in the product automata $(S1 \times S2)$ multiplied by the size of the longest non looping path multiplied by $(S1 \times S2)$(i.e cumulative credentials takes$(S1 \times S2)^3$) . As a result, the complexity for the algorithm will be ((T1 $\times$ T1) + $(S1 \times S2)^3$).

Example: In this example, web service business protocol performing two operations with two different ACP and client business protocol that interacts with this service are presented in figures 11 and 12 respectively. Figure 13 and 14 show the product automata of the two protocols and the graphical representation of the used ontology. Note that the compatibility of the two protocols partly depends on the subsumption relation between school and student cards.

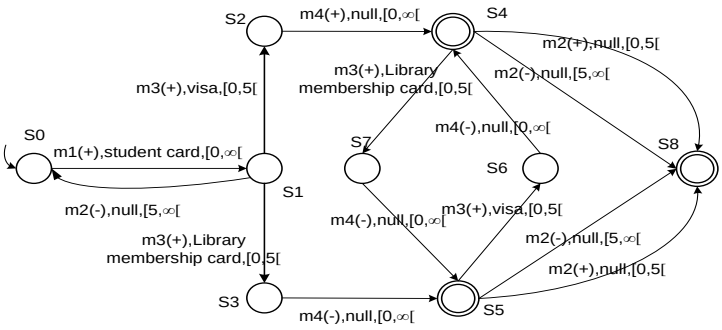

Fig. 11. Business protocol of web service performs two operations.

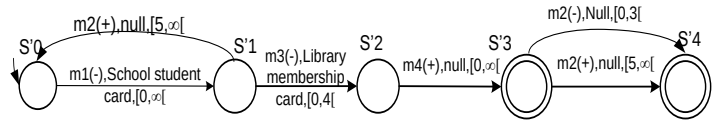

Fig. 12. Business protocol of a consumer needs to interact with the service in figure 11.

Algorithm 2: Compatibility between two protocols using cumulative product automata Part 1.

Input: $P^1(S^1, s_0^1, T^1, F^1)$ and $P^2(S^2, s_0^2, T^2, F^2)$
Output: checking compatibility result: The protocol P^1 and P^2 are compatible or not.
E_c //set of compatible states in s^p
E_{ca} // set of co-accessible states in E_c
C // set of all the paths in A^p
$Cmessage$ // boolean variable set to true if the two protocols are compatible in terms of message exchange and false otherwise
$modifiedEca \leftarrow true$ // Boolean variable used for verifying the co-accessibility, it is true each time Eca changed
$Ec \leftarrow (s_0^1, s_0^2)$
-Create the product automata annotated with ACP (the same messages with different polarities and the ACP ,credentials and policy, with polarity) product automata of P^1 and P^2, $A^p = P^1 \times P^2 = (S^p, S_o^p, T^p, F^p)$
-Checking the compatibility in terms of message exchanged using the created product automata.
//finding the couples of compatible states in S^p
foreach $(s_i^1, s_i^2) \in S^p$ **do**
  //verifying the output message from P^1
  if $\forall(s_i^1, s_{i+1}^1, m_i^-, c_i^1, I_i^1) \in T^1 \exists(s_i^2, s_{i+1}^2, m_i^+, p_i^2, I_i^2) \in T^2$ *where* $I_i^1 \subseteq I_i^2$ *and* $(s_i^1, s_i^2, s_{i+1}^1, s_{i+1}^2, \vec{m}_i, c_i^1$ *and* $p_i^2, I_i^1) \in T^p$ *and* $(s_{i+1}^1, s_{i+1}^2) \in R$ **then**
    Continue
  else
    Return false
  end
  //verifying the output message from P^2
  if $\forall(s_i^2, s_{i+1}^2, m_i^-, c_i^2, I_i^2) \in T^2 \exists(s_i^1, s_{i+1}^1, m_i^+, p_i^1, I_i^1) \in T^1$ *where* $I_i^2 \subseteq I_i^1$ *and* $(s_i^1, s_i^2, s_{i+1}^1, s_{i+1}^2, \overleftarrow{m}_i, c_i^2$ *and* $p_i^1, I_i^2) \in T^p$ *and* $(s_{i+1}^1, s_{i+1}^2) \in R$ **then**
    Continue
  else
    Return false
  end
end
$E_c \leftarrow E_c \cup (s_i^1, s_i^2)$
//verifying the co-accessibility of states in E_c
$E_{ca} = E_c \cap (s_F^1, s_F^2)$ **while** *modifiedEca* = *true* **do**
  modifiedEca = false
  forall $\in (s_i^1, s_i^2) \in E_c \notin E_{ca}$ **do**
    if $\exists(s_j^1, s_j^2) \in E_{ca}$ *and* $((s_j^1, s_j^2), (s_i^1, s_i^2), m_i, I_i^1) \in T^p$ **then**
      $E_{ca} \leftarrow E_{ca} \cup (s_i^1, s_i^2)$ modifiedEca $\leftarrow$ true
    end
  end
end

Algorithm 3: Compatibility between two protocols using cumulative product automata part 2.

if $E_c - E_{ca} \neq 0$ **then**
  Return false
else
  Cmessage =True
end
-Checking the compatibility in terms of ACP using the created product automata(checking the cumulative access control on the product automata).
c_i^1: cumulative credentials corresponding to protocol P^1 and assigned to the state s_i.
c_i^2: cumulative credentials corresponding to protocol P^2 and assigned to the state s_i.
c_{ij}^1: cumulative credentials corresponding to protocol P^1 and assigned to the transition between s_i and s_j(i.e union of set of credentials in those transitions.
c_{ij}^2: cumulative credentials corresponding to protocol P^2 and assigned to the transition between s_i and s_j.
foreach *state* $s_i \in output(s_0)$ **do**
  $c_i^1 = c_{0i}^1$
  $c_i^2 = c_{0i}^2$
  ENQUEUE(s_i, c_i^1, c_i^2)
end
while $Q \neq empty$ **do**
  $Temp\_Q = DEQUEUE(Q)$
  foreach $s_j \in output(s_i) in (s_i, c_i^1, c_i^2) = Temp\_Q$ **do**
    $c_j^1\_temp = c_j^1$
    $c_j^2\_temp = c_j^2$
    if $c_j^1 \neq null$ **then**
      $c_j^1 = (c_{ij}^1 \sqcap c_i^1) \sqcup c_j^1$
    else
      $c_j^1 = (c_{ij}^1 \sqcap c_i^1)$
    end
    if $c_j^2 \neq null$ **then**
      $c_j^2 = (c_{ij}^2 \sqcap c_i^2) \sqcup c_j^2$
    else
      $c_j^2 = (c_{ij}^2 \sqcap c_i^2)$
    end
    $c_{ij}^1 = c_{ij}^1 \sqcap c_i^1$
    $c_{ij}^2 = c_{ij}^2 \sqcap c_i^2$
    if $\neg((c_j^1 == c_j^1\_temp)$ *and* $c_j^1 \neq null$ *and* $(c_j^2 == c_j^2\_temp)$ *and* $c_j^2 \neq null)$ **then**
      ENQUEUE(Q, s_j, c_j^1, c_j^2)
    end
  end
end
- **if** $\forall p_i^1, c_i^1 \sqsubseteq p_i^1$ *in the cumulative product automata satisfying it and* $\forall p_i^2, c_i^2 \sqsubseteq p_i^2$ *satisfying it* **then**
  Return: The two protocols are compatible in terms of ACP.
else
  Return: The two protocols are not compatible in terms of ACP.
end

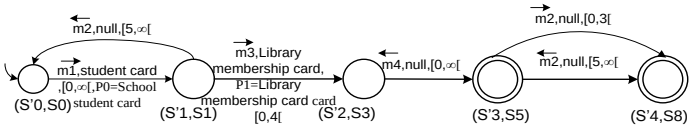

Fig. 13. Product automata of the two protocols of figures 11 and 12 assigned with ACP.

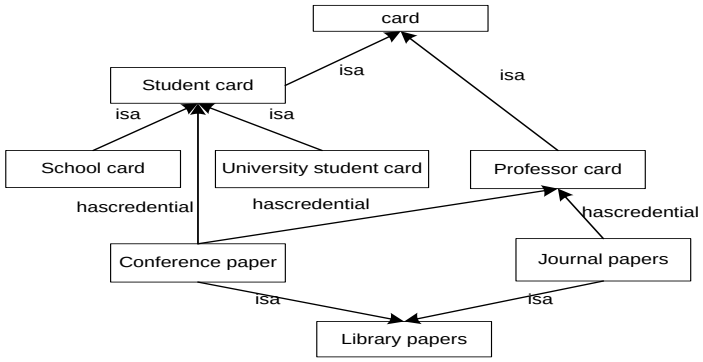

Fig. 14. Graphical representation of resources ontology linked with the credential ontology.

4.2 Replaceability

In our work we are interested in two types of replaceability analysis: full replaceability and replaceability in terms of interaction with specific consumer. Protocol P^1 can be fully replaced by protocol P^2 if and only if all the protocols that are compatible with P^1 are compatible with P^2. Protocol P^2 can replace P^1 in terms of interaction with consumer protocol P^3 if and only if protocol P^2 is compatible with P^3 which is compatible with P^1.

Definition 10. *The **intersection automata** A^i between two timed business protocols $P^1 = (S^1, s_0^1, T^1, F^1)$ and $P^2 = (S^2, s_0^2, T^2, F^2)$ is defined as $A^i = (S^i, s_o^i, T^i, F^i)$ where:*

- $S^i = S^1 \times S^2$
- $s_o^i = (s_0^1, s_0^2)$
- T^i *is the greatest subset of* $((S^1 \times S^2) \times (S^1 \times S^2) \times ((M^- \times 2^{c^1}) \cup (M^+ \times p^1)) \times I)$ *such that for for all transition* $((s_i^1, s_i^2), (s_{i+1}^1, s_{i+1}^2)), \overleftrightarrow{m}_i, p^p, c^p, I^p) \in T^p$ *there exist two transitions* $(s_i^1, s_{i+1}^1, m_i, (p^1 \text{ or } c^1), I^1) \in T^1$ *and* $(s_i^2, s_{i+1}^2, m_i, (p^2 \text{ or } c^2)), I^2) \in T^2$ *with :*

- $I^i = I^1 \cap I^2$
- *polarity*$(m_i, P^1) = polarity(m_i, P^2)$ *and*
  - *If polarity* $(m_i, P^{1,2}) = -$ *then* $c_i^i = c_i^1$ *and* c_i^1
  - *otherwise,* $(m_i, P^{1,2}) = +, p^i = p_i^1$ *and* p_i^1

– $F^i = F^1 \times F^2$

Definition 11. *(Cumulative path in the intersection automata):* $PI^p = ((s_i^1, s_i^2) \overset{(P_i^1, C_i^2)}{\rightarrow} (s_{i+1}^1, s_{i+1}^2), ..., (s_n^1, s_n^2) \overset{P_n^1, C_n^2)}{\rightarrow} (s_{n+1}^1, s_{n+1}^2)))$ *is a cumulative path in the intersection automata* $A^i = (s^i, s_o^i, Ti, F^i)$ *where*

- *States* $(s_i^1, s_i^2), ..., (s_n^1, s_n^2) \in S^i$
- *Each policy* P_i^1 *is the set of cumulative policies which is the union of the previous set of cumulative policies* P_{i-1}^1 *and the current policy* p_i^1 *where* p_i^1 *is the policy on the transition between the state* s_i^1 *and* s_{i+1}^1 *of the protocol* p^1 *and* $P_0^1 = p_0^1$ *and each cumulative credential* C_i^2 *is the set of cumulative credentials which is the union of the previous set of cumulative credential* C_{i-1}^2 *and the current set of credentials* c_i^2 *where* c_i^2 *is the set of credentials on the transition between the state* s_i^2 *and* s_{i+1}^2 *of the protocol* p^2 *and* $C_0^2 = c_0^2$.
- *A complete cumulative path in the intersection automata is the cumulative path which starts with the initial state* (s_0^1, s_0^2) *and ends with a final state* $(s_f^1, s_f^2) \in F^i$.

Definition 12. *(Replaceability in terms of intersection automata assigned with ACP) Protocols* $P^1 = (S^1, s_0^1, T^1, F^1)$ *and* $P^2 = (S^2, s_0^2, T^2, F^2)$ *are two Timed business protocol, and* $A^i = P^1 \bigcap P^2 = (s^i, s_o^i, T^i, F^i)$ *is a intersection automata assigned with ACP, we say that* P^1 *can be fully replaced by* P^2 *using their intersection automata if there is a relation* $R = S^1 \times S^2$ *where for all* $(s_i^1, s_i^2) \in R$*:*

- $\forall (s_i^1, s_{i+1}^1, m_i^+, p_i^1, I_i^1) \in T^1 \exists (s_i^2, s_{i+1}^2, m_i^+, p_i^2, I_i^2) \in T^2$ *where* $(s_i^1, s_i^2, s_{i+1}^1, s_{i+1}^2, \overleftarrow{m}_i, p_i^i, I_i^i) \in T^i$, $I_i^1 \subseteq I_i^2$ *and* $(s_{i+1}^1, s_{i+1}^2) \in R$
- $\forall (s_i^2, s_{i+1}^2, m_i^-, c_i^2, I_i^2) \in T^2 \exists (s_i^1, s_{i+1}^1, m_i^-, c_i^1, I_i^1) \in T^1$ *where* $(s_i^1, s_i^2, s_{i+1}^1, s_{i+1}^2, \overrightarrow{m}_i, c_i^i, I_i^i) \in T^i$, $I_i^2 \subseteq I_i^1$ *and* $(s_{i+1}^1, s_{i+1}^2) \in R$.
- $(s_{i+1}^1, s_{i+1}^2) \in s^i$*is co-accessible, there is a path in the intersection automata from this state to final state.*
- $(s_0^1, s_0^2) \in R$.
- *For all the complete cumulative paths* $PI^p = ((s_0^1, s_0^2) \overset{P_0^1, C_0^2)}{\rightarrow} (s_1^1, s_1^2), ..., (s_n^1, s_n^2) \overset{(P_n^1, C_n^2)}{\rightarrow} (s_{n+1}^1, s_{n+1}^2))$ *in the intersection automata, any set of credential satisfy* p_i^2 *can also satisfy the cumulative policy* P_i^1 *and* $c_i^1 \subseteq C_i^2$.
- P^1 *can be replaced by* P^2 *in terms of interaction with* $P^3(S^3, s_0^3, T^3, F^3)$ *if and only if* P^2 *is compatible with* P^3 *which is compatible with* P^1.

The algorithm for checking the replaceability uses the same mechanism which is used in the compatibility algorithm. The idea is to traverser through the intersection automata starting from the initial state and checking the mentioned properties of the relation R. This part is presented by algorithm 4. Algorithm 5 presented the second part which uses the same technique as algorithm 3 but instead of calculating the cumulative

credentials it calculates the cumulative policies for the first protocol P^1 and the cumulative credentials for the second protocol P^2. For all policies in the intersection automata, any set of credential satisfy p_i^2 can also satisfy the cumulative policy P_i^1 and the set of credentials c_i^1 is a subset of the set of cumulative credentials C_i^2. The complexity of this algorithm is the same as the complexity of the compatibility algorithm because they use the same mechanism but with different way of manipulation.

4.3 Proofs

A.Compatibility in terms of product automata implies compatibility in terms of interaction trace: Protocols $P^1(S^1, s_0^1, T^1, F^1)$ and $P^2(S^2, s_0^2, T^2, F^2)$ are two timed business protocol, and $A^p = P^1 \times P^2 = (s^p, s_o^p, T^p, F^p)$ is their product automata assigned with ACP, suppose we have $tr = (s_0^1, s_0^2, \overleftrightarrow{m}_1, t_1, s_1^1, s_1^2), (s_1^1, s_1^2, \overleftrightarrow{m}_2, t_2, s_2^1, s_2^2),$, $(s_{i-1}^1, s_{i-1}^2, \overleftrightarrow{m}_i, t_i, s_i^1, s_i^2)$ is a partial interaction trace. We have P^1 and P^2 are compatible in terms of automata so there is a relation $R = S^1 \times S^2$ where for all $(s_i^1, s_i^1) \in R$:

- $\forall(s_i^1, s_{i+1}^1, m_i^-, c_i^1, I_i^1) \in T^1 \exists(s_i^2, s_{i+1}^2, m_i^+, p_i^2, I_i^2) \in T^2$ where $I_i^1 \subseteq I_i^2$ and $(s_i^1, s_i^2, s_{i+1}^1, s_{i+1}^2, \overrightarrow{m}_i, c_i^1$ and $p_i^2, I_i^1) \in T^p$ and $(s_{i+1}^1, s_{i+1}^2) \in R$ and $c_0^1 ... c_i^1$ satisfy p_i^2 which implies that there is a new transitions is added to the partial interaction trace tr, this transition is $(s_i^1, s_i^2, \overrightarrow{m}_{i+1}, p_{i+1}^2$ and $c_{i+1}^1, I_{i+1}^1, s_{i+1}^1, s_{i+1}^2)$ and because s_{i+1}^1, s_{i+1}^2 is co-accessible(there is a path to the final state) then each interaction trace is included in a complete interaction trace ,(i.e. each interaction trace $IT(tr.((s_i^1, s_i^2, \overrightarrow{m}_{i+1}, p_{i+1}^2$ and $c_{i+1}^1, I_{i+1}^1,$ $s_{i+1}^1, s_{i+1}^2))$ can reach a final state). Therefore, each interaction trace belongs to IT starts by initial state and ends by final state so it is included in a complete interaction trace and $c_0^1 ... c_i^1$ satisfy p_i^2
- $\forall(s_i^2, s_{i+1}^2, m_i^-, c_i^2, I_i^2) \in T^2 \exists(s_i^1, s_{i+1}^1, m_i^+, p_i^1, I_i^1) \in T^1$ where $I_i^2 \subseteq I_i^1$ and $(s_i^1, s_i^2, s_{i+1}^1, s_{i+1}^2, \overleftarrow{m}_i, c_i^2$ and $p_i^1, I_i^2) \in T^p$ and $(s_{i+1}^1, s_{i+1}^2) \in R$ and $c_0^2 ... c_i^2$ satisfy p_i^1 which implies that there is a new transitions is added to the partial interaction trace tr, this transition is $(s_i^1, s_i^2, \overleftarrow{m}_{i+1}, p_{i+1}^1$ and c_{i+1}^2, I_{i+1}^1 $, s_{i+1}^1, s_{i+1}^2)$ and because s_{i+1}^1, s_{i+1}^2 is co-accessible(there is a path to the final state) then each interaction trace is included in a complete interaction trace ,(i.e. each interaction trace $IT(tr.((s_i^1, s_i^2, \overleftarrow{m}_{i+1}, p_{i+1}^1$ and $c_{i+1}^2, I_{i+1}^1, s_{i+1}^1, s_{i+1}^2))$ can reach a final state). Therefore, each interaction trace belongs to IT starts by initial state and ends by final state so it is included in a complete interaction trace and $c_0^2 ... c_i^2$ satisfy p_i^1

END Proof

B.Compatibility in terms of interaction trace implies compatibility in terms of automata Protocols $P^1(S^1, s_0^1, T^1, F^1)$ and $P^2(S^2, s_0^2, T^2, F^2)$ are compatibles in terms of interaction trace and $A_p = (s^p, s^{p, T^p, F^p)}$ is their product automata assigned with ACP.Suppose there is a relation R between each pairs of states of the two protocols such that $((s_i^1, s_i^2) \in R$: then $\forall tr \in IT(P^1, P^2)$, its last tuple $(s_{i-1}^1, s_{i-1}^1, \overrightarrow{m_{i-1}}$ $, c_{i-1}, t_{i-1}, s_i^1, s_i^2)$ verifies:

– $\forall m \forall t$ if $\exists(s_i^1, s^1, \text{M}^-, \text{C}^1, \text{I}^1) \in T^1$ such that the set of credentials c sent in m matches C^1, $t \in \text{I}^1$and m is an instance of M then $\exists s^2 \in S^2$ such that $tr.(s_i^1, s_i^2, \overrightarrow{m}, c, t, s^1, s^2) \in IT(P^1, P^2)$.

  Since $\exists$ transition $(s_i^2, s_{i+1}^2, m_{i+1}^+, p_{i+1}) \in T^2$ and $(s_i^1, s_i^2, \overrightarrow{m}_{i+1}, p_{i+1}andc_{i+1}$, $s_{i+1}^1, s_{i+1}^2) \in IT$, where IT is the set of all transition in the interaction trace. This result presents the first property of the relation R where (s_i^1, s_i^2) and $(s_{i+1}^1, s_{i+1}^2) \in R$. Since $(tr.(s_i^1, s_i^2, \overrightarrow{m}_{i+1}, p_{i+1}$ and $c_{i+1}, s_{i+1}^1, s_{i+1}^2)) \in IT$ is includes in a complete interaction trace, then (s_i^1, s_i^2) and (s_{i+1}^1, s_{i+1}^2) are co-accessible which is the second property. This interaction trace is starts by the initial state (s_0^1, s_0^2) and there is a path from it to the final state because tr is included in complete interaction trace so this state(s_0^1, s_0^2) is co-accessible which is the third property of the relation R.
– $\forall m \forall t$ if $\exists(s_i^2, s^2, \text{M}^-, \text{C}^2, \text{I}^2) \in T^2$ such that the set of credentials c sent in m matches C^2, $t \in \text{I}^2$and m is an instance of M then $\exists s^1 \in S^1$ such that $tr.(s_i^1, s_i^2, \overleftarrow{m}, c, t, s^1, s^2) \in IT(P^1, P^2)$. Since $\exists$ transition $(s_i^1, s_{i+1}^1, m_{i+1}^+, p_{i+1}) \in T^1$ and $(s_i^1, s_i^2, \overleftarrow{m}_{i+1}, p_{i+1}$ and c_{i+1}, s_{i+1}^1, $s_{i+1}^2) \in IT$, where IT is the set of all transition in the interaction trace. This result presents the first property of the relation R where (s_i^1, s_i^2) and $(s_{i+1}^1, s_{i+1}^2) \in R$. Since $(tr.(s_i^1, s_i^2, \overleftarrow{m}_{i+1}, p_{i+1}andc_{i+1}, s_{i+1}^1, s_{i+1}^2)) \in IT$ is includes in a complete interaction trace, then (s_i^1, s_i^2) and (s_{i+1}^1, s_{i+1}^2) are co-accessible which is the second property. This interaction trace is starts by the initial state (s_0^1, s_0^2) and there is a path from it to the final state because tr is included in complete interaction trace so this state(s_0^1, s_0^2) is co-accessible which is the third property of the relation R.

END Proof

5 Conclusion

In this paper we investigated a high-level analysis and management of business protocols of web services that is aware of access control policies assigned to service operations. Beside protocol annotation with policies and credentials, we defined notions of compatibility and replaceability based on those annotated protocols. Together with those notions, we proposed algorithms for checking compatibility between two services and for checking whether one service can transparently replace another from the point of view of compatibility.

The contributions of this paper can be extended in several directions. First, the time constraints considered here could be extended to consider timeouts w.r.t. several previous transitions as in [12]. An other direction for future work consists in abstracting annotations in order to deal with other functional and non functional properties, such as quality of service or privacy. A third possible extension of this work is to automatically build adapters allowing two services to work together even though they are not directly compatible.

References

1. Benatallah, B., Casati, F., Toumani, F., Ponge, J., Nezhad, H.: Service mosaic: A modeldriven framework for web services life-cycle management. IEEE Internet Computing **10** (2006) 55–63
2. COMPAS: (Compliance-driven models, languages, and architectures for services) http://www.compas-ict.eu/index.php.
3. Benatallah, B., Casati, F., Ponge, J., Toumani, F.: Compatibility and replaceability analysis for timed web service protocols. In: Procs of BDA, Saint-Malo, France (2005)
4. Berners-Lee, T., Hendler, J., Lassila, O.: The semantic web. Scientific American (may 2001) 34–43
5. Cardoso, J.: Semantic Web Services - Theory, Tools, and Applications. Idea Group (2006)
6. Alur, R., Dill, D.: Automata for modeling real-time systems. In: Proceedings of the 17th International Colloquium on Automata, Languages and Programming. Volume 443., Editors. Springer-Verlag (1990)
7. Larsen, K.G., Pettersson, P., Yi, W.: Uppaal in a nutshell. International Journal of Software Tools for Technology Transfer **1** (1997) 134–152
8. Benatallah, B., Casati, F., Toumani, F.: Web service conversation modeling: A cornerstone for e-business automation. IEEE Internet Computing **8** (2004) 46–54
9. Benatallah, B., Casati, F., Toumani, F.: Representing,analysing and managing web service protocols. Data and Knowledge Engineering **58** (2006) 327357
10. Benatallah, B., Casati, F., Toumani, F.: Analysis and management of web services protocols. In: Proceedings of ER, Shanghai, China (2004)
11. Baina, K., Benatallah, B., Casati, F., Toumani, F.: Model-driven web service development. In: Proceedings of CAiSE, Riga, Latvia (June 2004)
12. Ponge, J., Benatallah, B., Casati, F., Toumani, F.: Fine-grained compatibility and replaceability analysis of timed web service protocols. In: ER. (2007) 599–614
13. Bertino, E., Squicciarini, A.C., Paloscia, I., Martino, L.: Ws-ac: A fine grained access control system for web services. World Wide Web **9** (2006) 143–171
14. Ferraiolo, D., Sandhu, R., Gavrila, S., Kuhn, D., Chandramouli, R.: Proposed nist standard for role-based access control. ACM Transactions on Information and System Security (TISSEC) **4** (2001) 224274
15. Sandhu, R., Coyne, E.J., Feinstein, H.L., Youman, C.E.: Role-based access control models. IEEE Computer **29** (February 1996) 3847
16. Mitchell, J., Winsborough, W.: Design of a role-based trust-management framework. security and privacy. In: Proceedings IEEE Symposium. (2002) 114130
17. Moses, T.: extensible access control markup language (xacml) version 2.0. In: OASIS Standard, 200502. (2005)
18. Damianou, N., Dulay, N., Lupu, E., Sloman, M.: The ponder policy specification language. In: Workshop on Policies for Distributed Systems and Networks; Springer-Verlag LNCS, Bristol, UK (Jan. 2001) 18–39
19. IBM Systems, Microsoft, A.S., VeriSign: Web services policy framework,"http://www-106.ibm.com/developerworks/library/specification/ws-polfram". (2006)
20. Kagal, L., Finin, T., Joshi, A.: Fourth ieee international workshop on policies for distributed systems and networks. (2003)
21. Tonti, G., Bradshaw, J.M., Jeffers, R., Montanar, R., Suri1, N., Uszok1, A.: Semantic web languages for policy representation and reasoning: A comparison of kaos, rei, and ponder. In: Proceedings of the 2nd International Semantic Web Conference (ISWC), Springer-Verlag (2003)

22. Nejdl, W., Olmedilla, D., Winslett, M., Zhang, C.: Ontology-based policy specification and management. In: 2nd European Semantic Web Conference (ESWC). Volume 3532 of Lecture Notes in Computer Science , Springer., Heraklion, Crete, Greece (2005) 290–302
23. Baader, F., Calvanese, D., McGuineness, D., Nardi, D., Patel-Schneider, P. In:. Cambridge University Press (2003)

Algorithm 4: Replaceability between two protocols using intersection automata part 1.

Input: Protocol $P^1(S^1, s_0^1, T^1, F^1)$ and $P^2(S^2, s_0^2, T^2, F^2)$ and consumer protocol $P^3(S^3, s_0^3, T^3, F^3)$.

Output: type of replaceability or no replaceability

E_c //set of replaceable states in S^i of the intersection automata

E_{ca} // set of co-accessible states in E_c

C // set of all the paths in A^i

$Cmessage$ // boolean variable set to true if the two protocols are replaceable in terms of message exchange and false otherwise

$modifiedEca \leftarrow true$ // Boolean variable used for verifying the co-accessibility, it is true each time Eca changed

$Ec \leftarrow (s_0^1, s_0^2)$

-Create the Intersection automata annotated with ACP (the same messages with the same polarities) intersection automata of P^1 and P^2, $A^i = P^1 \bigcap P^2 = (S^i, S_o^i, T^i, F^i)$

-Checking the replaceability in terms of message exchanged using the created intersection automata. //finding the couples of replaceable states in S^i

foreach $(s_i^1, s_i^2) \in S^i$ **do**

  //verifying the output message from P^1

  if $\forall(s_i^1, s_{i+1}^1, m_i^+, p_i^1, I_i^1) \in T^1 \exists(s_i^2, s_{i+1}^2, m_i^+, p_i^2, I_i^2) \in T^2$ *where* $(s_i^1, s_i^2, s_{i+1}^1, s_{i+1}^2, \overleftarrow{m}_i, p_i^i, I_i^i) \in T^i$, $I_i^1 \subseteq I_i^2$ *and* $(s_{i+1}^1, s_{i+1}^2) \in R$ **then**

    Continue

  else

    Return false

  end

  //verifying the output message from P^2

  if $\forall(s_i^2, s_{i+1}^2, m_i^-, c_i^2, I_i^2) \in T^2 \exists(s_i^1, s_{i+1}^1, m_i^-, c_i^1, I_i^1) \in T^1$ *where* $(s_i^1, s_i^2, s_{i+1}^1, s_{i+1}^2, \overrightarrow{m}_i, c_i^i, I_i^i) \in T^i$, $I_i^2 \subseteq I_i^1$ *and* $(s_{i+1}^1, s_{i+1}^2) \in R$ **then**

    Continue

  else

    Return false

  end

end

$E_c \leftarrow E_c \cup (s_i^1, s_i^2)$

//verifying the co-accessibility of states in E_c

$E_{ca} = E_c \cap (s_F^1, s_F^2)$ **while** *modifiedEca = true* **do**

  modifiedEca = false

  forall $\in (s_i^1, s_i^2) \in E_c \notin E_{ca}$ **do**

    if $\exists(s_j^1, s_j^2) \in E_{ca}$ *and* $((s_j^1, s_j^2), (s_i^1, s_i^2), m_i, I_i^1) \in T^p$ **then**

      $E_{ca} \leftarrow E_{ca} \cup (s_i^1, s_i^2)$ modifiedEca $\leftarrow$ true

    end

  end

end

Algorithm 5: Replaceability between two protocols using intersection automata part 2.

Input: $P^1 = (S^1, s_0^1, T^1, F^1)$ and $P^2 = (S^2, s_0^2, T^2, F^2)$, product automata $A^p = P^1 \times P^2 = (s^p, s_o^p, T^p, F^p)$

Output: protocols P^1 and P^2 are compatible in terms of ACP or not.

- Calculate the cumulative credentials for the protocol P^2 and the cumulative ACP of the protocol P^1 on the automata.

p_i^1: cumulative ACP corresponding to protocol P^1 and assigned to the state s_i.

c_i^2: cumulative credentials corresponding to protocol P^2 and assigned to the state s_i.

p_{ij}^1: cumulative ACP corresponding to protocol P^1 and assigned to the transition between s_i and s_j(i.e union of set of credentials in those transitions.

c_{ij}^2: cumulative credentials corresponding to protocol P^2 and assigned to the transition between s_i and s_j.

foreach *state* $s_i \in output(s_0)$ **do**
- $p_i^1 = p_{0i}^1$
- $c_i^2 = c_{0i}^2$
- ENQUEUE(s_i, c_i^1, c_i^2)

end

while $Q \neq empty$ **do**
- $Temp\_Q = DEQUEUE(Q)$
- **foreach** $s_j \in output(s_i) in (s_i, p_i^1, c_i^2) = Temp\_Q$ **do**
  - $p_j^1\_temp = p_j^1$
  - $c_j^2\_temp = c_j^2$
  - **if** $p_j^1 \neq null$ **then**
    - $p_j^1 = (p_{ij}^1 \sqcap p_i^1) \sqcup p_j^1$
  - **else**
    - $p_j^1 = (p_{ij}^1 \sqcap p_i^1)$
  - **end**
  - **if** $c_j^2 \neq null$ **then**
    - $c_j^2 = (c_{ij}^2 \sqcap c_i^2) \sqcup c_j^2$
  - **else**
    - $c_j^2 = (c_{ij}^2 \sqcap c_i^2)$
  - **end**
  - $p_{ij}^1 = p_{ij}^1 \sqcap p_i^1$
  - $c_{ij}^2 = c_{ij}^2 \sqcap c_i^2$
  - **if** $\neg((p_j^1 == p_j^1\_temp)$ *and* $p_j^1 \neq null$ *and* $(c_j^2 == c_j^2\_temp)$ *and* $c_j^2 \neq null)$ **then**
    - ENQUEUE(Q, s_j, c_j^1, c_j^2)
  - **end**
- **end**

end

- **if** $\forall p_i^1$ *and* p_i^2*,in cumulative intersection automata, all the set of credentials that satisfy* p_i^1 *satisfy* p_i^2 *and* $\forall c_i^1$ *and* c_i^2*,* $c_i^1 \sqsubseteq c_i^2$ **then**
- **Return**: Protocol P^1 can be fully replaced by P^2 in terms of ACP.

else
- **Return**: Protocol P^1 can not be replaced by P^2 in terms of ACP.

end