Compliance of Web services over a high level specification

Emad Elabd

N° d'ordre 122- 2011 Année 2011

THESE DE L'UNIVERSITE DE LYON

Délivrée par

L'UNIVERSITE CLAUDE BERNARD LYON 1

ECOLE DOCTORALE

Infomaths

DIPLOME DE DOCTORAT

(arrêté du 7 août 2006)

Soutenue publiquement le (13 Juillet 2011)

par

Emad ELABD

TITRE :

Compliance of Web Services Over a High Level Specification

« Conformité de services Web par rapport à des spécifications de haut niveau »

Directeurs de thèse :

Mohand-Said HACID	Prof., Université de Lyon 1
Emmanuel COQUERY	Maître de Conférences, Université de Lyon 1

JURY

Rapporteurs :	M. Farouk TOUMANI	Prof., Université Blaise Pascale, France
	Mme. Fatiha ZAIDI	Prof., Université Paris-Sud XI, France
Examinateurs :	Mme. Daniela GRIGORI	Prof., Université de Versailles St-Quentin en Yvelines; France
	Mme. Salima BENBERNOU	Prof., Université Paris Descartes, France
Directeurs :	M. Mohand-Said HACID	Prof., Université de Lyon 1, France
	M. Emmanuel COQUERY	Maître de Conférences, Université de Lyon1, France

AUTHOR BIOGRAPHY

Name :	Emad Said ELABD
Occupation:	Assistant lecturer in Information System Department
Occupation Place:	Faculty of Computers and Information, Menoufia University
Date of Birth:	21 January 1978
First University Degree:	B.Sc. of Electronics Engineering, Computer Science and Engineering Department.
Grade:	Very Good with honor degree
Educational Institution:	Faculty Of Electronic Engineering, Menoufia University.
Date of Educational Degree:	May 2000.
Second University Degree	M.Sc. In Computers and Information, Information System Department.
Educational Institution:	Faculty of Computers and Information, Menoufia University.
Date of Educational Degree:	July 2006.
Ph.D Registration Date:	October 2007

ACKNOWLEDGMENT

First of all, I thank my God for helping me to achieve this work and giving me the ability to finish this thesis in that satisfactory form. Second, I would like to express my appreciation to my supervisor **Prof. Mohand-Said HACID** for his continuous support and encouragement during the research study in this thesis. He really influenced my way of thinking and developing the research ideas adopted in this thesis. I am very grateful for his strong effort with me and his highly useful advice throughout the development of this work.

Third, it is a great pleasure for me to express my sincere appreciation and my thanks to **Dr. Emmanuel Coquery** for his supervision and continuous encouragement and supporting throughout the whole period of this work. Special acknowledgment is given to him.

Finally, this work is especially dedicated to my family especially my father (**Said**), mother (**Ehitimad**), brother(**Abdel-aleem**), sisters, my wife **Nermine** and her dead father and her mother, my children **Minnat-Alla**, **Muhamad**, and **Ziyad** and all my colleges in the database team. Without their support, encouragement and patience I would not have been able to finish this work.

Emad Elabd

ABSTRACT

French

Actuellement, la technologie des services Web évolue rapidement, en étant soutenue par les grands acteurs du domaine des systèmes d'information. Les applications basés sur services Web sont faiblement couplées et utilisables de façon automatique via l'utilisation d'un ensemble de normes basées sur XML. Hormis la description syntaxique des messages, il est nécessaire d'avoir une description sémantique du comportement des services. En effet, lors de la conception d'un service ou lors d'une composition de services, il est important de vérifier la conformité avec un cahier des charges. L'enrichissement des descriptions des services par l'inclusion de leurs comportements est de plus en plus important. Ce comportement peut être décrit par des protocoles métier représentant les séquences possibles d'échanges de messages.

Les services Web utilisent des politiques de contrôle d'accès (ACP) pour restreindre l'accès à des consommateurs autorisés. Ces politiques doivent faire partie de la description du service. Dans cette thèse, l'analyse d'interopérabilité en termes de contrôle d'accès après la formalisation des services Web annotés avec les politiques de contrôle d'accès est réalisée. Nous présentons une approche pour intégrer les outils de vérification dans l'architecture de contrôle d'accès de façon à garantir une interaction sans erreurs. Les politiques et les crédits sont présentés comme une ontologie afin de bénéficier de la flexibilité offerte par subsomption sur des concepts.

La chorégraphie des services Web est utilisée dans la phase de conception d'applications pair à pair complexes dans lesquelles chaque pair peut être implémenté par un service Web. Par conséquent, la sélection des services Web pour l'implémentation de chorégraphie en utilisant l'approche de vérification de compatibilité avec contrôle d'accès est l'un des objectifs de notre recherche. Dans ce travail, les modèles de protocole métier du service Web sont étendus en ajoutant des informations au message sur chaque transition du service dans lequel ce message sera envoyé ou reçu. Nous définissons et vérifions la compatibilité des services Web afin de voir si (et comment) plusieurs services peuvent avoir des interactions en fonction de leurs protocoles. Cette approche aidera les concepteurs à choisir des services Web de manière simple et à vérifier s'ils peuvent mettre en œuvre la chorégraphie nécessaire en vérifiant la compatibilité avec notre approche.

En plus contrôle d'accès, le temps joue un rôle crucial dans de nombreux comportements des services Web. Une des principales contributions de cette

thèse consiste la modélisation et l'analyse de service Web avec contraintes de temps pour garantir la compatibilité et la remplaçabilité. L'un des verrous fondamentaux rencontré avant l'analyse de compatibilité est l'élimination des transitions implicites temporisées dans les protocoles métier. Nous présentions une approche pour éliminer ces implicites sans changer la sémantique de protocole. Après avoir enlevé les transitions implicites, nos algorithmes de compatibilité et de remplaçabilité peuvent travailler d'une manière simple.
Plus le service Web est complexe, plus les paramètres qui influent sur le comportement de ce service sont nombreux. Enrichissement du comportement des services Web par certains paramètres comme le temps et les politiques de contrôle d'accès peut être généralisé pour inclure d'autres paramètres tels que l'information sur la les données privées, la signification du message, etc. Dans ce contexte, l'une de nos contributions est de fournir un modèle général pour les protocoles métier annotés par des spécifications de messages. Chaque spécification de message contient les contraintes et les informations qui sont nécessaires au service fourni. Ainsi, l'algorithme de vérification de compatibilité traite avec tous les types de contraintes et chaque service peut trouver le service le plus compatible avec lui en termes des contraintes nécessaires et des valeurs fournies.
Pour conclure, cette thèse vérifie la conformité des services Web, y compris après l'inclusion d'un ensemble de contraintes telles que la politique de contrôle d'accès et de temps. Le formalisme des automates temporisés et les contraintes sont utilisées pour représenter la spécification des services pour lesquels la conformité des services doit être vérifiée. L'analyse de compatibilité et de remplaçabilité entre les services Web en utilisant leurs protocoles métier est réalisée en présence de ces contraintes. Des algorithmes sont alors développés pour la vérification de la conformité avec comme objectif d'assurer la conformité à ces spécifications de haut niveau.
Toutes ces techniques et algorithmes seront validés par leur intégration et leur utilisation dans la plate-forme ServiceMosaic et au sein du projet COMPAS.

Mots-clés. Services Web, des protocoles d'affaires, de compatibilité, interchangeabilité, contrôle d'accès, le temps, la chorégraphie, la spécification chronométrés message automates, l'architecture orientée services.

English

Currently, Web services technology is rapidly move forward supported by major players in the field of information systems. Web services applications are loosely coupled and usable in an automatic way via the use of a set of standards based on XML. Beside the syntactic description of messages, there is a need for the semantic description of the behavior of services. Indeed, whether in the design of a service or composition of services, it is important to check compliance with a set of specifications. Enriching services descriptions by including their behaviors is becoming more and more important. This behavior can be described by business protocols representing the possible sequences of message exchanges.

Web services use access control policies (ACP) to restrict the access to authorized consumer. These policies should be a part of the service description. In this thesis, the interoperability analysis in terms of AC after the formalization of the Web services annotated with the access control (AC) is performed. In addition, we present an approach for embedding the checking tools in the AC enforcement architecture to guarantee the errors free interaction. The ACP and the credentials are presented as ontology in order to benefit from the flexibility offered by subsumption on concepts.

Web services choreography is used in the design phase of complex peer-to-peer applications in which each peer can be implemented by a Web service. Therefore, selecting Web services for choreography implementation using the compatibility checking approach with access control is one of the objectives of our research. In this work, the business protocol models of the Web service are extended by adding information to the message on each transition about the service in which this message will sent to or received from. We define and verify Web service compatibility in order to see if (and how) several services can have interactions based on their protocols. This approach will help the designers to select Web services in an easy way and verify if they can implement the required choreography or not by checking the compatibly using our approach.

In addition to AC, time has a crucial role in many of Web services behavior. Therefore, modeling and analyzing Web services based on error free compatibility and replaceability checking with time constraints is one of our major contributions in this thesis. One of the fundamental challenges before checking the compatibility and replaceability between timed business protocols is the removal of the implicit

transition of the timed business protocols without changing the semantics of the protocol. Therefore, we present a general approach for removing any form of implicit transition without changing the semantics of the protocol. After removing the implicit transitions, our compatibility and replaceability algorithms can work in a straightforward way.

The more the Web service is complex, the more it has parameters that affects the behavior of this service. Enriching the Web service behavior by certain parameters such as time and ACP can be generalized to include any other parameter such as privacy information, message meaning, etc. In this context, one of our contributions is to provide a general model for Web service business protocol annotated by message specifications. Each message specification contains the constraints and the information that are required of provided by the service. Thus, the compatibility checking algorithm deals with all the types of constraints and each service can find the most compatible service with it in terms of the required constraints and the provided values.

To conclude, this thesis checks the compliance of Web services after including a set of constraints such as the access control policy and time. The formalism of timed automata and constraints are used to represent the services specification for which the compliance of the services must be verified. Compatibility and replaceability analyses between the Web services using their business protocols are performed in the presence of these constraints. Algorithms are then developed for the verification of compliance with the aim of ensuring compliance with these high level specifications. All of these techniques and algorithms will be validated through their integration and use in the platform ServiceMosaic and in the COMPAS.

Keywords. Web Services, Business protocols, Compatibility, Replaceability, Access control, time, choreography, timed automata, message specification, Service-oriented architecture.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

Abbreviation	Referenced Terms
2PC	Two-Phase Commit
ASM	Abstract State Machine
AC	Access Control
ACES	Access Control Enforcement System
ACP	Access Control Policy
ACC	Access Control Credentials
API	Application Program Interface
B2B	Business-to-Business
BP	Business Protocol
BPEL	Business Process Execution Language
CACC	Cumulative Access Control Credentials
CORBA	Common Object Request Broker Architecture
CWS-RBAC	Composite Web Services-Role Based Access Control
CGI	Common Gateway Interface
DLs	Description Logics
EAI	Enterprise Application Integration
FO	First Order
GCBP	General Constraints Business Protocol
HTTP	Hypertext Transfer Protocol
IT	Information Technology
LTL	Linear Temporal Logic
MCETBP	Multi-Clocks Explicitly Time Business Protocol
MCTBP	Multi-Clocks Timed Business Protocol
OCTBP	One-Clock Timed Business Protocol with Implicit Transitions.
OCETBP	One Clock Explicitly Time Business Protocol
OWL	Ontology Web Language
PAP	Policy Administration Point
PEP	Policy Enforcement Point

PDP	Policy Decision Point
PIP	Policy Information Point
PS	Policy Selection
QoS	Quality of Service
RBAC	Role Based Access Control
SAML	Security Assertion Markup Language
SAWSDL	Semantic Annotations Web Service Description Language
SKU code	Stock-Keeping Unit
SLA	Service Level Agreements
SOA	service-oriented architectures
SOAP	Simple Object Access Protocol
SMTP	Simple Mail Transfer Protocol
SPOCUS	Semi Positive Cumulative State
SWS-RBAC	Single Web Service-Role Based Access Control
SSL	Secure Sockets Layer
TBP	timed business protocol
TLA	Trustworthiness Level Assignment
UDDI	Universal Description, Discovery and Integration registry
URL	Uniform Resource Locator
UPC	Universal Product code
W3C	World Wide Web consortium
WS-	Web Service-
WSCL	Web Services Conversation Language
WS-CDL	Web Services Choreography Description Language
WSCI	Web Service Choreography Interface.
WSDL	Web Service Description language
WSTL	Web Service Transition Language
XML	eXtensible Markup Language
XACML	eXtensible Access Control Markup Language

NOMENCLATURES

Nomenclature	Referenced Terms
A^p	The product automata of two business protocols
AR	The atomic role on the ontology
CN	The atomic concepts on the ontology
CR	The set of clock reset in the BP
CRS(s)	Clocks Reset Sequence of the state *s* in BP
CRV	Clock reset value
CV	Clock value
c_i^1	The set of credentials on the transition between the state (s_i^l and $s_{i+1}{}^l$) of the BP P^l
C_i^1	The set of cumulative credentials on a transition *i* on the BP P^l
ds	The destination state of a transition in the BP
E	The specification attribute domain
F	The set of final states in the BP
I	The set of time intervals
IP	The implicit path in the BP
IT	The interaction trace in the BP
k	A variable obeying to the sequence of natural numbers
L_i	The number of the implicit transitions in the BP
L_e	The number of explicit transition in the BP
M	The set of input/output messages
m	An instance message in the BP
ms	Message specification
MS	The set of message specification
mt	The message type
Na	The state name in the BP
o	The operation of the p_d that invoked by p_s and triggers the message exchange
PA	Path in the BP

p_d	The business protocol of the receiver of the message
PL	The set of access control policies
PLP	The Path-Loop-Path structure in the BP
p_s	is the business protocol of the message s sender,
$\mathbb{R}$	The set of real numbers.
RK	The set of variables reset on the BP
s	A state in the BP
S	The set of the states in the BP
S_0	The start state of the BP
s_f	a final state in the BP
ss	Source state of a transition in the BP
SV	Specification value
T	A finite set of explicit transitions in BP
Tc	The set of the time constraints on BP
Tc(ti)	Time constraint on an implicit transition *ti*
T_i	A set of implicit transition
tr	Time trace
TS	Non deterministic transition system
X	A finite set of clocks on the BP
x_g	The global clock in the one clock BP
xm	An xml message before annotation
xms	An xml message sequence of messages before annotation
XM	An annotated xml message after annotation
XMS	An annotated xml message sequence after annotation
XML	The set of all message in xml format
σ	A sequence of timed BP

Chapter 1 . INTRODUCTION

This chapter presents an overview of the thesis. It starts by explaining the reasons behind choosing the Web services as an effective solution for the business process integration problems. After that, we discuss our main research objectives and contributions. Finally, the outline of this thesis is listed.

1.1 Background

In the beginning of the Web, the Web users were restricted by static Web pages in which users cannot affect on the contents of these pages. After that, a new kind of pages allows the visitors not only to passively read the information, but also to interact and modify this information is emerged. For example, commercial services, such as airplane ticket reservations, hotel booking, etc., are offered via the Web. Most of these services are based on a business-to-customer communication where customers access the information systems via customer-oriented Web interfaces. For instance, an interface for books store is appeared on the Web browser as a set of visual tools such as buttons and lists etc. in order to ease the selection and buying of the books. Thus, most of the companies adopt the Web-based communication with customers. But, if we talk about the business-to-business communication, we will find that this type of communication tends to be less Web-enabled than business-customer communication (Rutner et al., [122]). Thus, the business-to-customer connection is established between the information system and a Web browser (e.g., Firefox and Internet Explorer) but business-to-business connections need integration operations between the information systems. Solving the integration problem between information systems is very costly using the traditional application integration middleware (e.g., the Remote Procedure Call (RPC) and messaging systems) because of the adapter development process.

Web services are loosely coupled applications designed to support interoperable machine-to-machine interaction over a network. Thus, Web services can be one of the solutions for this integration problem. The information systems are provided as Web services with standard interfaces. This standardization that will be in terms of description languages, coordination, and interaction protocols, will simplify the integration process between the information systems (Papazoglou and Georga-

kopoulos [108]; Papazoglou et al. [109]; Papazoglou and van den Heuvel [110]). Therefore, Web services are adapted as a framework for business-to-business interaction (Alonso et al. [4]).

Web services are loosely coupled applications. This means that when the service interface is developed, there is no information about the client interface in which the service interacts with. They could interact with many clients and the client must be aware of the functional and non functional properties of the service to interact correctly with it. As a result, the information that is included in the service interface is not sufficient for the client to check if he/she can interact correctly with the service before interaction and there is a need for more information. This type of information is called service description. Service description doesn't not only describe the interface, but also the business protocol (BP) of the services by representing the possible sequences of message exchanges (Benatallah et al. [16]).

Recently, Web services protocol modeling and management gain a great importance. The existing tools that model Web service protocols such BPEL (Business Process Execution Language) present service descriptions without studying the interoperability properties (Alves et al. [6]). These interoperability properties include checking the compatibility, by checking if two services can interact correctly or not?, and the replaceability, by checking if one of the two services can replace the other one or not?, between the Web services. Many approaches have been developed in the direction of formal methods and software tools for modeling and analyzing Web services protocols (Pong et al. [114]; Pong [113]; Benatallah et al. [19, 20, 18] ; Dumas et al. [61]; Ramsokul et al. [117] ; Hamadi et al. [79]). They presented a model for business protocol and a framework for protocol-based analysis.

1.2 Research objectives

Web service can be simple or complex depends on its functionality. The behavior of a Web service is affected by a set of parameters depends on the function of the service. The types and the number of these parameters are different from one service to another. One of these important parameters is the access control (AC) which includes the access control policy (ACP) and the access control credentials (ACC) of the Web service. Since a lot of Web services use access control policies to restrict the access to authorized consumers, these policies should be a part of the

service description. Security technologies commonly adopted for Web sites and traditional access control models are not satisfactory (Bertino et al. [29]). 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 role based access control (RBAC) model (Ferraiolo et al. [67]), WS-AC1 (Bertino et al. [29]), and conversation-based Web services access control model by (Mecella et al. [100]). In a parallel, researchers have developed policy languages for access control. These include eXtensible Access Control Markup Language (XACML) (Tim Moses [101]), WS-Policy (Bajaj et al. [11]) and finally to semantic Web based languages such as Rei (Kagal et al. [85]) and KAoS (Tonti et al. [131]).

The majority of the current and future generations of Web services need a conversation-based Web services access control models. This is because the Web service consists of a set of ordered operations and each operation could have an AC. The service consumer must respect the order of the operations and by consequence the order of the AC on these operations to interact correctly with the services. To the best of our knowledge, the current conversation-based Web services access control models such as the (Mecella et al. [100]) model does not perform interoperability and integration analysis in terms of AC. As a result, this kind of models does not guarantee an error free interaction between the services in the runtime because they did not perform any interoperability checking between the services before their interaction in the design time. These interoperability checking includes the compatibility checking using the business protocols annotated with the AC. Thus, modeling and analyzing Web service after including the AC is one of the objectives of this work.

There is a set of challenges raises during the analysis of the Web services business protocols after including the AC. One of these significance challenges is the calculation of the cumulative access control credentials (CACC) on each transition on the protocol (i.e., the calculation of the previous and the current credentials that are annotated on the protocol on each transition).

Web services choreography is used in the design phase of complex peer-to-peer applications in which each peer can be implemented by a Web service. The behavior of each peer must be specified in the choreography of the application. Any Web service that would like to join the choreography would need to conform to that

specification. Several research efforts focus on the issue of determining whether the behavior of the Web services implementing choreography matches the one described by the choreography specification (Paci et al. [107], Busi et al. [43], Kazhamiakin et al. [89]). Behavior conformance must include the satisfaction of the AC between services. Therefore, selecting Web services for choreography implementation using the compatibility checking approach with access control is one of the objectives of our research.

The second parameter that has a crucial role in many of Web services behavior is the time. Time-related behaviors can be session timeouts or deadlines with different kinds of behaviors (e.g., the situation where Visa card must be provided within *n* hours, otherwise the service will be cancelled). Recently, there are many research works concentrate on time as an effective player on Web services behaviors. For instance, the work presented by (Pong et al. [113, 114]; Benatallah et al. [17]; Berardi et al. [23]; Dyaz et al. [62], and Kazhamiakin et al. [88]). Some of these works involve timing issues in the Web service behavior which is presented by the business protocol (Pong et al. [113]). They perform nontraditional checking analysis (compatibility and replaceability analysis) on business protocols. But their compatibility and replaceability definition cannot be used in our analysis, for two reasons. First, our definitions of compatibility and replaceability checking are based on the error free interaction which is different from their definitions which are based on the language inclusion. Second, the technique which they use with the implicit transitions on the business protocol (i.e., the internal transition of the Web service) is different from our approach. Their technique does not work with a wide range of Web services business protocols. For example, the protocols which have loops and one of the transitions on the loop is an implicit transition with time constraint. Furthermore, the implicit transition on the loop has a clock reset and the time constraint on the implicit transition checks this clock.

As a result, modeling and analyzing Web services based on error free compatibility and replaceability checking with time constraints is one of our research objectives in this thesis.

The more the Web service is complex, the more it has parameters that affects the behavior of this service. Enriching the Web service behavior by certain parameters such as time and AC can be generalized to include any other parameter such as privacy information, message meaning, etc. Therefore, the message on each transition in the protocol can include a set of constraints and each constraint can be satis-

fied be set of values of its type. In this context, one of our objectives is to provide a general model for Web service business protocol annotated by message specifications. Each message specification contains the constraints and the information that are required of provided by the service. Performing high level analyses between Web services after enriching their behavior by adding the AC, time, etc., faces a set of difficulties and challenges.

1.3 Contributions

In this thesis, we present formal modeling and analysis of Web services behaviors. Compatibility and replaceability analyses between the Web services business protocols are performed. In this context, a set of concepts and techniques for performing compatibility and replaceability analysis between Web services using their business protocols is provided. These concepts include the formal definitions of the business protocols, product automat of two protocols, intersection automat of two protocols, compatibility and replaceability between services using their business protocols. All of these definitions and approaches are applied on the business protocols annotated with AC, time, and message specifications. After adding the AC, time, etc, the definition of the resulted business protocol is a modified form of the definition that is provided by (Benatallah et al. [19, 20, 18], Ponge et al. [113, 114]).

Checking the compatibility between Web services with AC faces some challenges, such as the calculation of the cumulative access control credentials on the transitions. Therefore, we present an algorithm for calculating the cumulative access control credentials on each transition by determining the current and the previous set of credentials that can be sent by services when it reaches this transition (Elabd et al. [63]).

One of the steps of the compatibility and replaceability checking algorithm is the comparison between the access control policy and the provided credentials. To accomplish this step correctly, the ACP and the credentials are presented as ontology 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. This contribution enables us to make use of the reasoning power of the ontology tools to determine the satisfaction between the provided set of credentials and the ACP.

Modeling Web service with time constraints is one of our major contributions. Timed Web services can have one clock or more depends on the function of the service. This thesis provides a set of algorithms for analyzing the business protocols with one clock or set of clocks (i.e., multi-clocks protocols where the clocks on the time constraints on any transitions are not necessary the clocks of the previous transitions but any clock for any transitions).

One of the fundamental challenges before checking the compatibility and replaceability between timed business protocols is the removal of the implicit transition of the timed business protocols without changing the semantics of the protocol. We provide two algorithms for removing the implicit transitions from timed business protocol. One for one clock business protocol and another one of multi clock time protocols. The conversion process is a complex task and depends on the implicit transition constraints, the clock resets on it, and if this transition is included in a loop or not. For instance, the simplest conversion examples are those for which implicit transitions do not have a clock reset and the hardest examples are those for which implicit transitions have an implicit clock, a time constraint using this clock and are included in a loop. Therefore, we present a general approach for removing any form of implicit transition without changing the semantics of the protocol. After removing the implicit transitions, our compatibility and replaceability algorithms can work in a straightforward way.

After modeling and analyzing Web service with Ac and time, it is important to generalize the approach to include any type of constraints on the services behaviors. Therefore, we presented the analyses of Web service behaviors after annotated the protocols with what is called "message specification". Thus, the compatibility checking algorithm deals with all the types of constraints and each service can find the most compatible service with it in terms of the required constraints and the provided values.

Another major contribution is the selection of Web services for choreography implementation using the compatibility and replaceability checking approach with AC (Elabd et al. [64]). In this part, the business protocol models of the Web service are extended by adding information to the message on each transition about the service in which this message will sent to or received from. We define and verify Web service compatibility in order to see if (and how) *n* services can have interactions based on their protocols. This approach will help the designers to select Web

services in an easy way and verify if they can implement the required choreography or not by checking the compatibly using our approach.

1.4 Outline of the Thesis

This thesis consists of the following chapters:

Chapter 1: Introduction. This chapter gives a generic view of the thesis by presenting a brief background about the subject of the research, the objectives of it, and our contributions.

Chapter 2: Web services. This chapter presents the definitions, architecture, semantics, behavior description, and modeling of Web services. The first section presents the various definitions of the Web services. The second section presents an overview of the conventional middleware and the service oriented architecture. The third section discusses the semantic Web and the semantic Web services. Finally, The Web service behavior description and modeling is discussed.

Chapter 3: Timed Web services. This chapter discusses modeling and analysis of timed Web services. It starts by giving an overview about timed Web services with concrete examples. Then, the compatibility and replaceability checking definitions are explained. This chapter discusses in details modeling one-clock and multi-clocks timed services. In addition, the problem of the implicit transitions in the analysis process is presented. The last section lists the related work on modeling timed Web services.

Chapter 4: Implicit transition removal approach. This chapter presents the conversion approach of business protocols after removing the implicit transitions without changing the semantics of the business protocols. It is divided to main parts; the first part shows the separation approach and the second part shows the main conversion approach.

Chapter 5: Web services access control: This chapter presents a set of Web services security concepts and shows the approaches in which these security requirements are modeled and analyzed. Firstly, it explores the Web service access control models. Secondly, an informal scenario and the proposed architecture are explained. Thirdly, the role of the ontology in the analysis is clarified. Finally, the related work are listed and discussed.

Chapter 6: Web services analysis. This chapter discusses in details the interoperability analysis between Web services. It starts by explaining the different definitions of compatibility and replaceability between services in details. These analyses use the Web services business protocols for presenting the behavior of the services. The next parts of this chapter provide the Web services analysis after including the AC and time constraints

Chapter 7: General specification approach. In the previous chapters, we discussed the modeling and analyzing of Web services interoperability in the presence of some important parameters such as time and AC. This chapter presents a general approach in which we can perform our analysis in the presence of many parameters. As much as the number of the parameters increases, the complexity increases and in some cases will be undicidable. Therefore, this chapter proposes a fined-grained compatibility checking approach.

Chapter 8: Web Services Choreography: This chapter shows an approach for selecting the Web services from the Web for implementing choreography for a complex process using compatibility and replaceability checking with access control. The verification process of the selected services is presented with the aid of an informal scenario.

Chapter 9: Conclusion and perspectives: This chapter presents the conclusion of this work and the proposed future work.

Chapter 2 . WEB SERVICES

Day after day Web services become a good proposer as a standard middleware between business-to-business applications. This chapter gives an overview of the Web services and how they can be used as a middleware. It starts by listing the various definitions of the Web services. Then, the service oriented architecture is presented. After that, the semantic Web architectures and the semantic Web services are discussed. The last two sections in this chapter describe the Web services behavior description and an overview of the Web services formalization and analysis, respectively.

2.1 Web Service definitions

Since the emergence of Web services, they have many definitions range from the very generic to the very specific and restrictive. For example, defining Web service as an application accessible to other applications over the Web is a very generic definition (see e.g., Alonso et al. [4]). This definition is very open because it includes all the application that has a Uniform Resource Locator (URL) (e.g., it includes the Common Gateway Interface (CGI) scripts and all the programs accessible on the Web with a stable application program interface (API) which are published with additional descriptive information on some service directory). Thus, this definition is not precise enough.

The World Wide Web consortium (W3C, 2002)[1] extended the previous definition by another accepted definition for Web service states that "*a software application or component that can be accessed over the internet using a platform/language-neutral data interchange format to invoke the service and supply the response, using a rigorously defined message exchange pattern, and producing a result that is sufficiently well-defined to be processed by a software application.*"

[1] http://www.w3.org/2002/ws/arch/2/wd-wsawg-reqs-03042002

A more precise definition concentrates on the technical details of the Web service is provided by the (W3C, 2004)[2], specifically the group involved in the Web Service Architecture group: *A Web service is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically, the Web service description language WSDL (Christensen et al. [48]). Other systems interact with the Web service in a manner prescribed by its description using* SOAP[3] *messages, typically conveyed using the Hypertext Transfer Protocol (HTTP) with an* XML[4] *serialization in conjunction with other Web-related standards* (McCabe et al. *[98]).*

The previous W3C definition is a very precise definition because it defines the Web service and shows how it works. It defines the Web services as software applications that are described using the WSDL and interacts with other services using the SOAP messages. This definition states that the XML is a part of the solution.

The Universal Description, Discovery and Integration registry (UDDI) consortium defined the Web services as *self-contained, modular business applications that have open, Internet-oriented, standards-based interfaces* (Bellwood et al. [15]). This is a precise definition and declares the standardization of the application interface as a major requirement. Furthermore, the service should be available over the Web and can be invoked using its standard interface. But this definition does not clarify enough what is meant by a modular, self contained business application.

IBM[5] presents a more precise definition of the Web service states that *Web services are a new breed of Web applications. They are self-contained, self-describing, modular applications that can be published, located, and invoked across the Web. Web services perform functions, which can be anything from simple requests to complicated business processes. Once a Web service is deployed, other applications (and other Web services) can discover and invoke the deployed service.* This definition shows the three main operations that could be applied to the Web services (*published, located, and invoked across the Web).* But it does not

[2] http://www.w3.org/TR/ws-gloss/
[3] Simple Object Access Protocol, http://www.w3.org/TR/soap/
[4] The Extensible Markup Language (XML) is a W3C Recommendation - http://www.w3.org/XML/
[5] http://www.redbooks.ibm.com/abstracts/sg246292.html

mention the standardization technology which is a main characteristic of the Web services.

There is even more specific definition of Web services. For instance, the definition on the Webopedia[6], *"a standardized way of integrating Web-based applications using the XML, SOAP, WSDL, and UDDI open standards over an Internet protocol backbone. XML is used to tag the data, SOAP is used to transfer the data, WSDL is used for describing the services available, and UDDI is used for listing what services are available"*. This definition concentrates on the Web services standard technologies and the integration ability which is a main target behind the use of the Web services.

To conclude, the web services definitions can be categorized into two categories, the high-level definitions and the technical definitions. In the high-level definitions, they concentrate on the paradigm of the SOA, the interoperable framework for message based and loosely-coupled interaction between software components and the platform independent implementations (see UDDI and IBM definitions). In the technical definitions, they concentrate on the use of the technologies such as the XML, SOAP protocols, WSDL for service description, and UDDI for registering the service (see Webopedia and W3C definitions).

2.2 Service Oriented architecture

The initial propose of the Web was to publish information using Web pages and this information could be accessed in a reliable and simple way by consumers. These Web pages are linked with each other and can be easily accessed and browsed by the users around the world. As a result, the Web is used by the originations to manage, organize, and distribute their internal data to consumers and partners. During that time period, the initial technologies associated with the Web were not sufficient enough to implement business-to-customer and e-commerce solutions. Therefore, additional functionality solutions were developed such as the SSL (Secure Sockets Layer) protocol that was developed for transmitting private documentation via the Internet. With the wide use of the Web, the number of customers and business partners for real-time information increases. Thus, the organi-

[6] http://www.webopedia.com/TERM/W/Web_services.html

zations were required to link their heterogeneous, autonomous and distributed systems to improve productivity and efficiency. This led to the development and deployment of EAI (Enterprise Application Integration) solutions.

Incompatible and distributed systems were integrated using EAI platforms. The problem was that many EAI frameworks required costly and proprietary protocols and formats with technical difficulties when it was needed to integrate internal systems with external systems running on partner's computers. To overcome the problem of integration between internal and external information systems, business-to-business (B2B) solutions were developed. Business processes between organizations were carried out more efficiently using the B2B infrastructures. Most of B2B solution relied on the use of the XML as a language to represent data.

There are certain application integration scenarios that are costly when they are performed by the B2B middleware. For example, two companies can use a centralized middleware for the B2B integration. This middleware is controlled by one of the two companies and use a platform accepted by the two parties (e.g., this platform uses a specific message broker, a specific workflow system, and a specific name and directory server). Indeed, the centralized middleware hosted by one of the participating companies or by a third party is not a preferable approach because of the lack of trust between companies, the autonomy that each company wants to preserve, and the confidentiality of the business transactions.

The mentioned limitations of the centralized B2B middleware can be overcome by the using a point-to-point technique where the two parties agrees on using certain middleware protocol and infrastructure hosts by each one (i.e., there is no third parties). In practice, the company can integrate with more than one company; therefore, it requires the use of different middleware platforms. Since there are many middleware in the company, the cost of the integration between it and other companies becomes high.

Consequently, organizations realized that their B2B strategies have led the development of architectural solutions that often exhibited a tight-coupling among interacting software applications which limited the flexibility and dynamic adaptation of Information Technology (IT) systems.

The Web paved the way toward facilitating application integration and brought set of standard such as standard interaction protocols (HTTP) and data formats (XML) that were adapted by many companies. Therefore, the Web presents the idea of common middleware infrastructures that reduce the heterogeneity among interfaces and systems.

As a result, Web services are nominated to overcome the limitations of the business-to-business integration conventional middleware. Nowadays, most enterprises use Web services as a framework for facilitating application-to-application interaction within and across them. There are three main aspects of Web service that push toward resolving the limitations of conventional middleware: service-oriented architectures, redesign of middleware protocols, and the standardization. Table 2-1 summarize the evolution of information systems integration with the new technologies.

Table 2-1: The evolution of information systems integration.

Architecture	Integration	Technology
Web	Publish information	Html
B2C	Perform transactions	SSL
EAI	Integrate internally	Propriety protocols
B2B	Integrate externally	XML
SOA	Universal integration	Web services

Web services technology is emerging as main pillar of service-oriented architectures (SOA) (Papazoglou et al. [110]). This technology facilitates application integration by enabling programmatic access to applications through standard XML-based languages and protocols. The service-oriented architectures (SOA) address the requirements of the loosely coupled distributed information systems. An SOA approach solves many problems between the distributed enterprises information systems such as application integration, transaction management, and security policies (Alonso et al. [4]). Therefore, Web services in SOA can be shared and reused (Kreger [91]). These facilities make the Web services indispensable for applica-

tions in the same enterprise or in different enterprises (Arsanjani [8]). In the SOA, the services are presented as self-contained software modules. These services are described using a standard Web service description language and provide a business functionality. Web services use set of standards such as the WSDL, SOAP, and UDDI. The main objective of a service in an SOA is to represent a reusable unit of business-complete work. The service in the SOA is self-contained (i.e., the service maintains its own state), platform independent (i.e., the service interface is not constructed to work on a specific platform), and can be located, invoked and (re-) combined (Papazoglou et al. [110]). Web services designers and developers create autonomous and independent Web services which are different from the exciting software integration solution such as the Common Object Request Broker Architecture **(**CORBA) which develops the interacted component with the same team.

The Web service is characterized by its interface and its implementation. The service interface is the part of the service which is seen by other services and contains the invoke information. The service implementation performs the function of the service and the service implementation is not shown to the other services. Therefore, the services that are collected for business processes are independent of the platform. The invocation information is always available for internal invocation or external invocation.

The service requester and the service provider communicated using messages formatted according to the simple object access protocol (SOAP). The procedure starts with a request from the service consumer by sending a SOAP message to the service provider. This message can be carried using the internet transport protocols such as the hypertext transfer protocol (HTTP) and the Simple Mail Transfer Protocol (SMTP). On the service provider side, the SOAP message is received by a SOAP listener that extracts the body of the message, transforms the XML message into native protocol, and delegate the request to the actual business process within an enterprise. The implementation of the service can be hosted within a Web services container (Dhesiaseenlan et al. [60]). The service containers help in the invocation, location, routing, and management of the services. After the execution of the service, the service provider sends back his response to the service requester in the form of the SOAP envelope containing the XML message.

Figure 2-1 shows an SOA and an SOA request-response pattern where the services registry (e.g., UDDI) services as intermediary that is interposed between service requesters and service providers. The service providers publish the definitions of the service they offer using the WSDL and the service requester find this information by search the registry.

Figure 2-2 shows the web service stack in which the first layer is the transports that uses the standard HTTP protocol. This stack shows that the web services use the XML as a standard message format and as a metadata description format. It shows also that the SOAP message should be secure, reliable, and transacted.

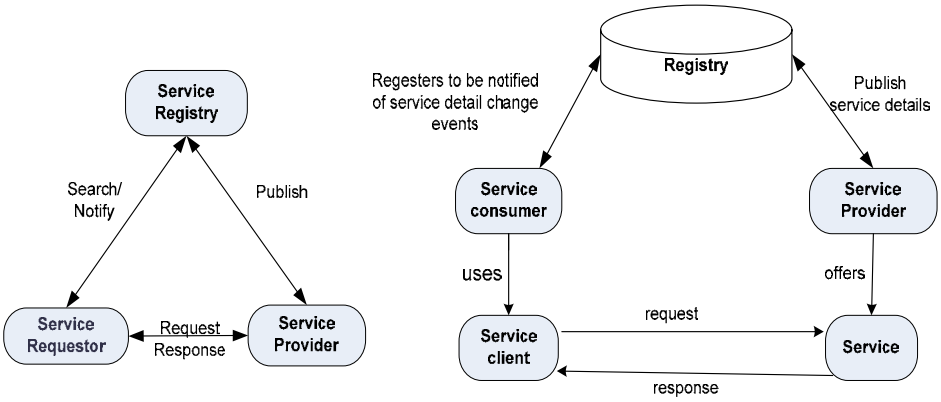

Figure 2-1: An SOA and an SOA Request-Response pattern with a service registry.

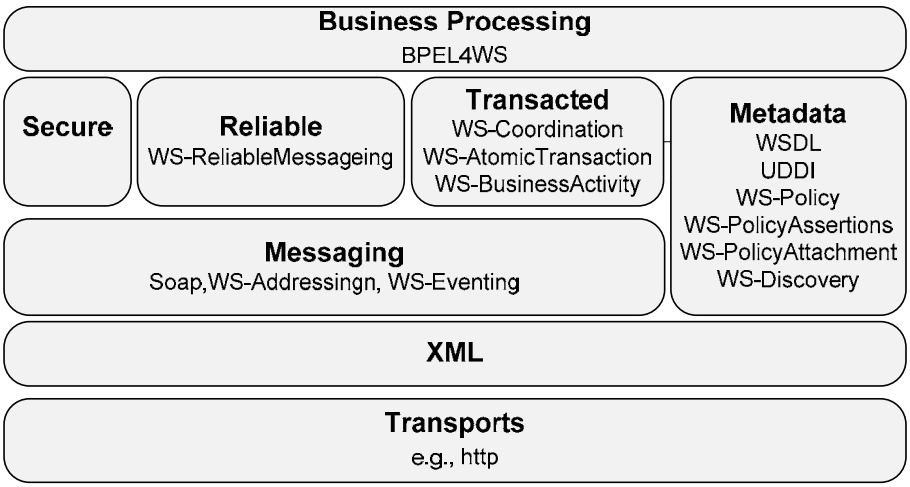

Figure 2-2: Web service Stack

Web service specifications are referred to as "WS-*" (Motahari et al. [105]). There is a long list of "WS-*" specification such as WS-Addressing, WS-Security, WS-Notification, WS-Transfer, WS-Eventing, and WS-Enumeration. Some of these specifications are not necessary, because of the probable overlap and inconsistencies among competing specifications that address the same functional areas (Vinoski [140]). For example, the WS-Eventing specification that is published by a group included Microsoft that aims to support event-based Web services has an overlap with the WS-Notification that is published by a group included IBM in the conceptual level but different on the detailed level (Vinoski [139, 138]).

The protocols which are used in the conventional middleware and works with central transaction coordinator, such as the Two-Phase Commit (2PC) protocol, are not able to work with peer-to-peer fashion. Therefore, the function that is achieved by these protocols in the centralized platform must be achieved by new protocol works in a decentralized setting and across trust domain (Alonso et al. [4]). Standardization in terms of the interface definition language and interaction protocols is the key towards the adaptation of Web services as a promising integrating solution between heterogeneous information systems applications.

2.3 Semantic Web

Nowadays, there are a lot of research efforts with the objective of adding more meaning to Web content in order to have what is called semantic Web (Fensel et al. [66]). Tim Berners-Lee et al. stated that "*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*" [28].

The semantic Web architecture consists of a series of standards organized into a certain structure express the interrelationships between these standards. The first diagram for the semantic Web architecture was presented by (Tim Berners-Lee et al. [28]). Figure 2-3 illustrates the different versions of the semantic Web reference architecture (V1-V4) proposed by (Berners-Lee et al. [25, 26, 27, 24], Gerber et al. [75]).

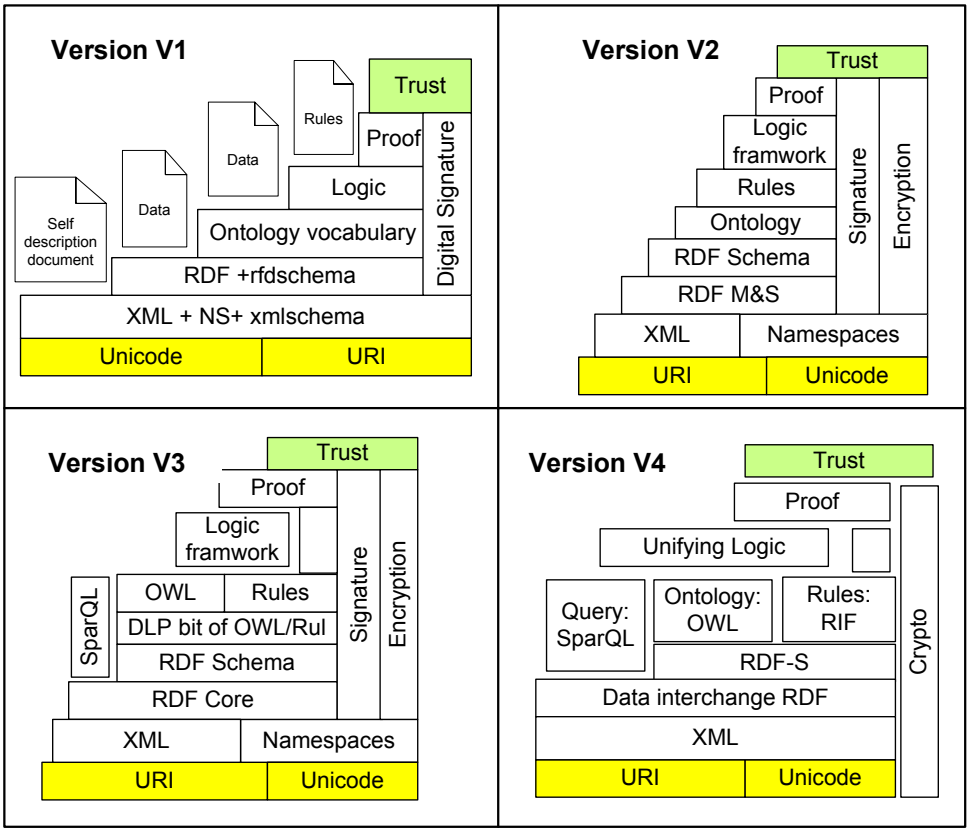

Figure 2-3: The four versions of the semantic Web reference architecture (V1-V4) proposed by Berners-Lee [25, 26, 27, 24, 75]

2.4 Semantic Web services

Several researchers have recognized that Web service standards lack of semantics (Cardoso [45]). Semantic description allows 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 services in integrating enterprise systems.

Semantic Web services are Web services whose properties, capabilities, interfaces, and effects are encoded in an unambiguous and machine-interpretable form (Mcilraith et al. [99]). The lifecycle of Web services can be divided to modeling, build-time and deployment, and run-time. Semantics have an important role in the com-

plete lifecycle of Web services (Sheth et al. [125]). For example, in the modeling phase, the Web service can be annotated by some information to explain the goal of the service by the service provider. The terms which are used in the description will be taken from a semantic model. After publishing the service in a registry, the service requester can search for a service using terms from the semantic model and reasoning techniques. As a result and due to the semantic model, there is an agreement on the meaning and there will be less ambiguity in the intended semantics of the provider.

Semantics Web service paves the way towards the automatic discovery of the Web service. This automatic discovery involves finding a service that matches a given set of functional and nonfunctional requirements in a repository of services. This match includes the syntactic and/or semantic matching. There are four scenarios after checking the matching between two services in terms of their syntax and semantics. The first scenario, the two services are fully matched because they are similar in their syntax and semantics. The second scenario, the two services are poorly matched because they are similar in their syntax and different in their semantics. The third scenario, the two services are poorly matched because they are similar in their semantics and different in their syntax. The fourth scenario, the two services are not matching because they are different in their semantics and syntax.

Semantics could help in the service invocation by providing a more detailed level of matching to identify the actual interface mappings. For example, if there is a service *A* that takes a "UPC code" as input and this service is matched semantically with another service *B* that provides "SKU code". However, there are differences in the syntactic representation of these two codes (e.g., UPC may be a 14 digit code while SKU is a 12 digit code). These differences information could be provided to the service *B* before invocation to perform the conversion process itself or using another conversion service. These types of information can be derived from a semantic model and mappings can be generated and stored to facilitate invocation.

Many business domains require composing multiple services to deliver new functionality. The semantics Web service is important in the composition process because it determines whether this composition gives the intended functionality or not by checking the non functional aspects of the composed services. The select and compose of services is based on the syntax and the semantic description of the

requirements and capabilities of the services. There are a lot of work is done to compose Web services based on nonfunctional properties by modeling the quality of service properties as constraints in mathematical programming (Zeng et al. [144]; Agarwal et al. [3]).

In the run time phase of web service lifecycle, semantics is important in situations that require automatic service discovery and binding to find suitable substitutable services in case of Web service failure. Web processes or complex Web services in the distributed interacting systems are modeled using the Finite State Machine (FSM) (Arthur Gill [76]) and its variants such as Petri nets (Wolfgang Reisig [118]), process algebra (Wan Fokkink [68]), situation calculus (Levesque et al. [94]) and *Pi* Calculus (Davide Sangiorgi [124]). The semantic Web services representation languages includes the languages for formal representations of ontologies (e.g., the description logics (Baader et al. [9]), frame logics (Kifer et al. [90]) and logic programming (Baral et al. [14])).

To summarize, finding and integration between business applications whose functionality is semantically described are more easily than those without semantic descriptions. This presents opportunities for semantic Web services in integrating enterprise systems. Therefore, augmenting the Web service description with nonfunctional properties such as the access control policy requirements is a contribution towards semantic Web services.

2.5 Web service behavior description

There is a set of standards that are used in the Web services description. The first standard is the XML which is adopted and commonly accepted as a standard common based language because its syntax is flexible enough to enable the definition of service description languages and protocols. The second standard is the Web services description language which is used for describing the Web services interfaces. WSDL is an XML vocabulary. It allows service authors to provide the essential information about the service so that others can use it. A WSDL document can be divided into two parts: a reusable abstract part and a concrete part. The abstract part of WSDL describes the operational behavior of Web services by describing the messages that go in and out of services. The concrete part of WSDL describes how and where to access the service implementation. The description of the semantic of

the service is not included in the WSDL document because it provides the syntactic or structural term for the messages that go in and come out from a service. It does not provide information on what the semantics of that exchange are.

Simple interface description is not always enough in Web services interaction. The only type of services where the interface description is sufficient is the Web services that offer only one operation. In this type of Web services, the service is invoked once and the consumer gets the result in one step (i.e., he/she sends a request and gets a response). But, this is not always the case of Web services. Most of Web services offer a set of operations that consumers must invoke in a certain order to achieve their functions. Such exchanges between the consumers and the Web services are called conversations. Therefore, there is a need for describing the behavior of the Web service by providing the set of message exchange between the Web service and the consumer with the order of sending or receiving of these messages. This set of message exchange rules is specified by the business protocol supported by the service. This protocol model is presented using state chart which is a suitable model for describing behaviors.

Figure 2-4 shows an example of a business protocol of an authentication Web service based on a specific identification card. In this model, states represent the various stages that a service may go through while transitions are triggered when a message is received or sent (e.g., examples of states include **Start**, **Fail**, **Identification**, and **WaitID** and examples of messages include **Opensession**, **IDrequest**, **WaitID**, and **IDchecking**). The positive polarity indicates that the message is an incoming message and the negative polarity indicates that the message is an outgoing message. There is a unique initial state (e.g., **Start**) and one or more final states (e.g., **Access** and **Fail**).

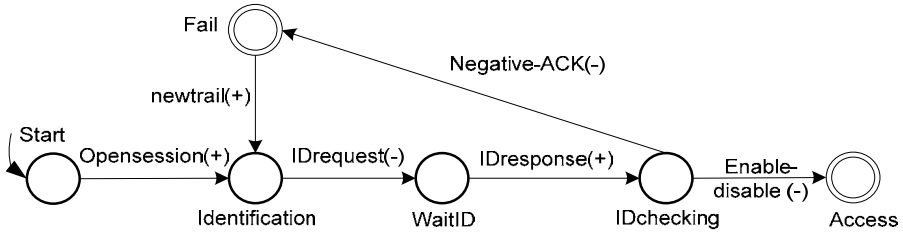

Figure 2-4: Authentication Web service business protocol.

Indeed, the complete description of the service includes not only its interface but also the business protocols that the Web service supports. There is a set of languages that can be used for defining business protocols such as Web Services Conversation Language (WSCL) (Banerji et al. [13]) and Business Process Execution Language for Web services (BPEL) (Alves et al. [6]).

Description can also include non-functional properties of the service such as Quality of Service (QoS) (i.e., network performance and reliability characteristics), cost of service, and Service Level Agreements (SLA). This information is important for the consumer in helping him to take the decision of using the Web service or not. This information includes the cost of service, the quality of service, and a textual description of the service. Constraints over the service behavior are considered as non-functional descriptions (Fensel et al. [49]). For example, in a hotel booking Web service, invoking its functionality (booking a room in a hotel) might be constrained by using a secure connection (security as non-functional property) or by actually performing the invocation of the services in a certain point in time (temporal availability as non-functional property). The non-functional properties can be used in the discovery, selection and substitution of services. Non-functional properties suffer from the lack of support in terms of the languages, methodologies, and tools due to various factors (Rosa et al. [120]; Christopher Van et al. [133]). These factors include the complexity of modeling and the difficulty of formalization of the non-functional properties. In addition, the non-functional properties are always presented informally in an abstract way and sometimes conflicts and compete with each others. Therefore, modeling Web services description with non-functional properties become a fundamental challenge in service oriented architecture especially in a real business setting.

In Web services, such information can be attached to the description of a service by using the repositories that the Web service publishes their services in it such as the UDDI. This information is registered with the service providers and queried by the consumers. Consumers could search for a service (Service discovery) during design-time, by browsing the directory and identifying the most relevant services, and at run-time, using dynamic binding techniques.

In this thesis, we concentrate on the Web service access control policy and time constraints as a part of the non-functional service description. After studying the

influence of including the AC and the time, a general approach is presented to show the effect of including other parameter in the service description. Therefore, we will study the business protocol of the Web service after including the AC and the time.

2.6 Web service formalization and analysis

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. There are three common model families. The first family is the activity based models which are used for representing systems in an executable form (e.g. workflow management systems (Van der Aalst [134])). The second family is the rule-based models (Forgy et al. [69, 70]) which define behaviour through a set of rules. The third family is the state-based models which mainly used for describing behavioral abstractions of a system. For instance, the UML models family (Rumbaugh et al. [121]).

Because state-based model is commonly used to model the behaviour of systems, due to the fact that it is simple and intuitive, we choose it in our model. Activity-based models are more suitable for creating executable models. Finally, rule-based models are a natural fit for complex decision-making systems where the logic must be frequently updated (e.g., by business analysts, accountants, etc.). They are however less suitable for describing behavior. Timed automata (Alur et al. [5]) are well known formalisms for real-time systems and there are some well-known tools supporting them as UPPAAL (Larsen et al. [92]). Therefore, it can be used for describing and analyzing the behavior of Web services, specifically those including time restrictions.

Various Web services models have been proposed for capturing different types of abstractions. For instance, (Dirk et al. [32]) define protocol interface formalism for services which is similar to the timed model proposed by (Berardi et al. [23]) but without time aspects. Bultan et al. [40] present a modelling for Web services interactions and present further discussion in [72, 42]. They present formalism for

specification and verification of electronic services for composition purposes. The set of services (peers) used in the model to present the different parts of the composition service.

Web services choreography and multi-party protocols formalism have been studied also. For instance, Qiu et al. [116] propose a language for Web services choreographies called **Chor** as a simplification of the Web services choreography description language (WS-CDL) (Ritzinger et al. [87]) (the reference specification for choreographies). Kang et al. [86] presents some tools enable WS-CDL for facilitating development of SOA systems. BPEL also offers abstract process for describing the externally observable behavior of a service composition. Van der Aalst et al. [136, 135] present YAWL, a general-purpose workflow language that has support for Web services. Papazoglou M. et al. [96] have done some work on the formalization of multi-party protocols with temporal constraints for service networks. Some work consider timing abstractions have been done by Kazhamiakin et al. [88] which mainly reuses well-known timed automata model-checking techniques in service-based compositions.

Nowadays, databases provide the backbone for a wide range of electronic commerce applications. Modeling database-driven systems are based on extending the classical model checking to infinite-state systems. There is some research works show that the data is the source of infinity. For instance, petri nets with data associated to tokens (Lazic et al. [93]), rewriting systems with data (Bouajjani et al. [35, 36]), automata and logics over infinite alphabets (Bouyer et al. [38, 37], [104], Demri et al. [57], Jurdzinski et al. [84], Bojanczyk et al. [33], Bouajjani et al. [36]), and temporal logics manipulating data (Demri et al. [57], [58]).

There are some modeling approaches model and verify the data which are stored in the database and can be used during the running of the business process in its model. Relational transducer is an example of these modeling approaches (Abiteboul et al. [1, 2]). They specified business models as relational transducers that map sequences of input relations into sequences of output relations. They present a restricted model, called a Semi Positive Cumulative State (SPOCUS) transducer. Their work addresses three main issues, temporal properties verification, log validation, and the comparison between two relational transducers. This relational transducer model has the ability to keep the history of the events in the system too.

This modeling approach can be used in cases where the historical data that is provided during the interaction is needed to be known in each state of the service. One example of these data is the ACP and the credentials that can be provided during the interaction. They defined the compatibility between two business process that there exists a run which achieves some desired goals while satisfying both business models. For unrestricted relational transducers, problem such as the compatibility is undecidable (Abiteboul et al. [2]). However, the restricted use of data and the particular properties verified have limited applicability to database-driven systems. In particular, model checking Linear Temporal Logic (LTL) properties in the presence of data quickly becomes undecidable (Vianu [137]).

Spielmann [128] presents the business model as abstract state machine (ASM) relational properties and perform the temporal properties verification, Log validation, and the comparison between two relational transducers. He proves that these problems are decidable under some restrictions. If one of these restrictions is satisfied then the problem is decidable. These restrictions are

- All (static) database relations of a relational transducer are known
- The maximal input flow which a relational transducer is exposed to be a priori limited

Deutsch et al. [59] extended the ASM transducer with two features: (i) the ability to constrain inputs by a First Order (FO) formula, and (ii) allowing access to previous user inputs. They call the new transducer ASM^{+}.

To summarize, modelling and analyzing Web service behaviour is an important issue specially after enriching the description of the service by including the AC and time parameters. Thus, in the next chapter we present our contributions with respect to the study of the Web service business protocols and the compatibility and replaceability analysis after including a set of parameters.

Chapter 3 . TIMED WEB SERVICES

This chapter discusses modeling and analysis of timed Web services. It starts by giving an overview about timed Web services with concrete examples. Then, the compatibility and replaceability checking definitions are explained. This chapter discusses in details modeling one-clock and multi-clocks timed services. In addition, the problem of the implicit transitions in the analysis process is presented. The last section lists the related work on modeling timed Web services.

3.1 Timed Web services

For many real business processes temporal restrictions are essential. Temporal constraints on Web services appear in many situations such as time-out for receiving a message from the consumer or performing an operation. For instance, some Web services are made up of sessions having an associated time-out, as those that allow a user to check her/his bank account or to participate in an online auction (i.e., if an operation does not take place on a given time interval then its execution on a further time can be insignificant). Therefore, incorporating time in Web services that implement these business processes became a necessity. These time constraints specification should be included in the Web services description in order to allow the consumer to check if he can correctly interact with theses services or not.

Modeling timed Web services is a major step towards their automated analysis. It eases the integration process by checking the interoperability properties between timed services. Web service behaviors can be modeled by business protocols representing the possible sequences of message exchanges.

Figure 3-1 shows an example of a business protocol of books selling Web service that has time constraints. The service starts by receiving a login message **(Login(+))**, then a message contains the selected product **(ChoosePRO(+))** from the consumer. The first time constraint is appeared on the transition between the state *S3* and *S5* (***T2=5* min**) which means that the consumer must get the price before five minutes after his/her selection of the book, otherwise he/her can cancel the selected book **(Cancel(+))** and login again. If the consumer received the price message before the five minutes, then he/she will send a delivery request (**DeliveryREQ(+)).** The confirmation of the delivery message will be sent to the con-

sumer based on the type of the delivery. If the consumer chooses the DHL, then the confirmation message must be send to him/her before three days **(DHL(-), *T5* < 3 days)** and if it is Normal, he/she will receive the confirmation message before five days **(Normal(-), *T5* <5 days)**. Otherwise, if six days pass before receiving the product, then the consumer can send failure message to renew the delivery request. After the consumer got the confirmation delivery message, he will send another confirmation message (**confirm(+))** to the Web service.

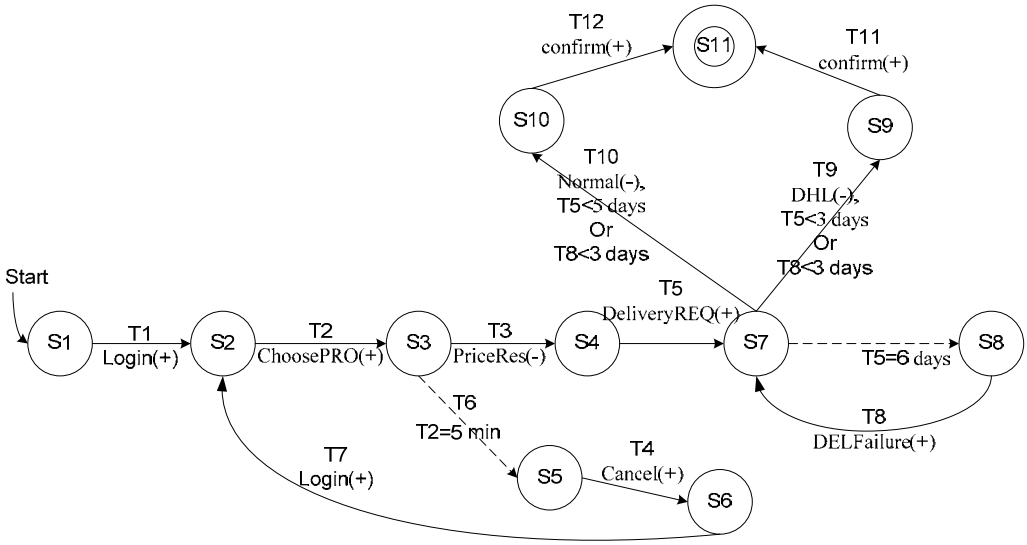

Figure 3-1: A Timed business protocol of book selling Web service.

There are two types of transitions in timed business protocols, explicit transition which expresses the change of the service from one state to another state according to an interaction with it by sending or receiving a message (e.g., the transition between the states *S1* and *S2* that has the **Login(+)** message), and implicit transition which is triggered by timeout and expresses the change from one state to another without an interacting action with the service (i.e., there is no message sent or received, e.g., the transition between the states *S3* and *S5* with the time constraint ***T2*=5 min**). Based on this definition of the explicit transition, the messages are presented with the explicit transitions only, and the implicit transitions are attached with time constraints. Time in this model is presented by clocks in which each clock is related to a transition. The number of clocks is based on the number of transitions which are used in time constraints by checking its time of triggering.

For instance, the clock *T2* is related to the transition between the states *S2* and *S3*, and reset when this transition is triggered. The implicit transition between the states *S3* and S5 has the constraint *T2*=5 min, which means that this transition must be triggered when the value of the clock *T2* equals to five minutes after the service entering into the state *S3*. There is another type of time constraints on the explicit transition which restrict the triggering of the transition to a specific time windows. For instance, the time constraint (*T5* < 3 days or *T8* <3 days) on the transition between the states *S7* and *S9* means that the **DHL(-)** message can be sent only in the time interval from zero to three days after the triggering of the transitions between the states *S4* and *S7* or the transition between the states *S8* and *S7* respectively.

3.2 Compatibility and replaceability

Compatibility and replaceability analysis have been discussed in some recent works [19, 20, 114, 34]. Bordeaux et al. in [34] present three different definitions for compatibility: (a) two services A and B are compatible if they have opposite behaviors; (b) two Web services are compatible if they do not have unspecified reception, and (c) two services are compatible if there is at least one execution leading to a pair of final states. There is a drawback in the first and in the second definitions, that is they do not check whether the interaction will reach a final state or not. The drawback of the third definition is that one execution does not guarantee an error free interaction. The source of these errors is the incompatible possible paths of the interacted service. In other words, during the interaction between two services, we do not know which paths will be taken, the paths that produce errors of the paths that do not make errors). The reason of the incompatibility in theses paths is the messages which have unspecified reception. Therefore, we present a new definition of compatibility (Elabd et al., [63]) by merging the second and the third definition of Bordeaux et al. [34]. In our definition, two services are compatible if and only if any potential message sent from one service can be received by the other service during their interaction and vice versa, and any reachable state is not in a deadlock, i.e., there is at least one execution leading to a pair of final states. Based on this definition, if two protocols are compatible, we guarantee that no error can happen during the interaction.

A model for business protocols and a framework for protocol-based analysis had been presented by Benatallah et al. [19, 20, 114]. This model captures all the conversations that are supported by a service. They studied the compatibility and re-

placeability issues. According to their definitions of compatibility, a Web service (*ws1*) with a business protocol *P1* is fully compatible with a Web service (*ws2*) with a protocol P2, if all the executions of *P1* can interoperate with *P2* and if only some of the executions of *P1* can interoperate with *P2*, then *ws1*is said to be partially compatible with *ws2*. The drawback of full compatibility is that one of the two protocols (e.g., *P1*) accepts all the execution of the other protocol (e.g., *P2*) but protocol *P2* may not accept all the executions of protocol *P1* and this produces errors during the interaction.

Example 3.1: Figure 3-2 shows an example of two business protocols *P1* and *P2*. The compatibility checking using the definitions of Benatallah et al. [16, 19, 20, 114] shows that the Web service (*ws1*) presented by the protocol *P1* is fully compatible with the Web service (*ws2*) presented by the protocol *P2* because all the conversation that can be established by the service *ws1* can be understood by the service *ws2* but the service *ws2* is partially compatible with *ws1* because there is at least one conversation can be established. Based on our definition of compatibility, we say that the two protocols are not compatible because the protocol *P2* shows that there is a potential message **CloseSession(-)** that can be sent by the service *ws2* and the protocol *P1* shows that the other service *ws1* will not able to receive this message when the two services are in the state *s2* and *s2* respectively. As shown in Figure 3-2 , there is also time constraints on each protocol ($X1 \in [0,10]$ in transition between the states *s1* and *s2* in *P1* and $X1 \in [0,5]$ in transition between the states *s1* and *s2* in *P2*). The time interval values of the two constraints implies that the time interval of the sent message **IDrequest(-)** is included in the time interval of the received message, and as a result, there is a satisfaction in terms of time constraints because the allowed time period for the sent message is included in the allowed time period of the received message.

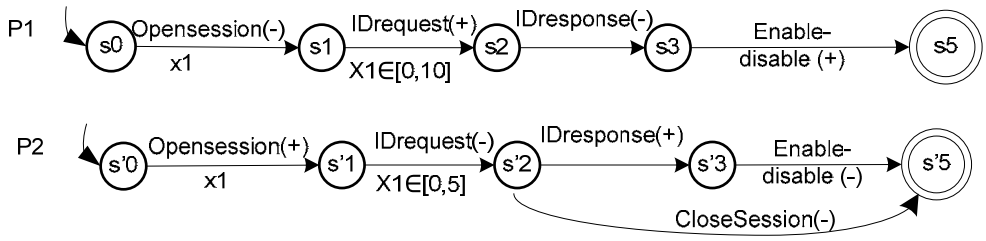

Figure 3-2: Two business protocols incompatible based on our compatibility definition and fully compatible based on the definition of Benatallah et al. [16, 19, 20, 114].

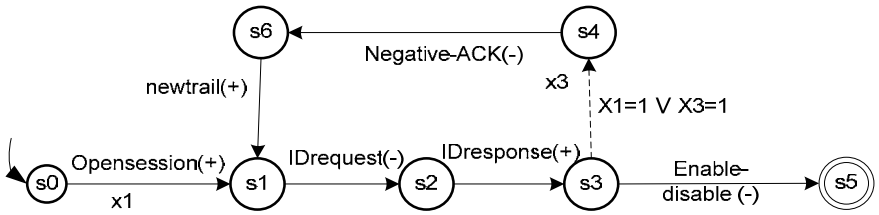

Figure 3-3: Authentication Web service business protocol with implicit transition on a loop and time constraint checking the implicit clock.

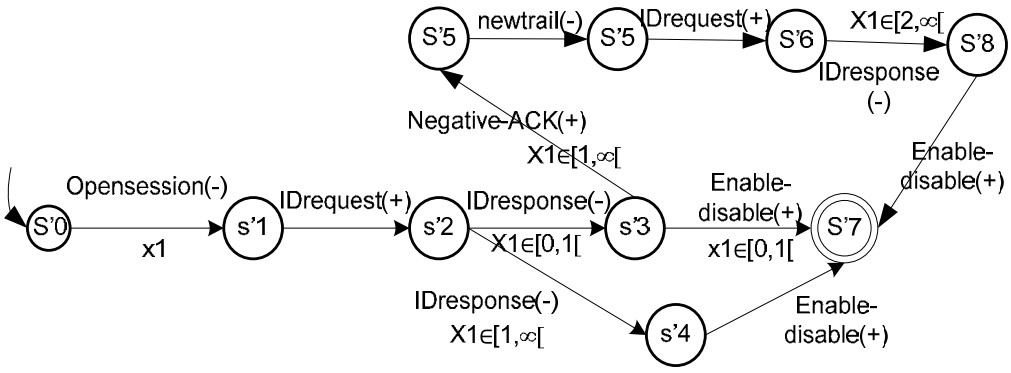

Figure 3-4: Business protocol without implicit transition.

3.3 Implicit transition issue

Example 3.1 shows that if the time constraints are presented explicitly on the transitions using time intervals, we can directly determine the satisfaction between the corresponding transitions based on the direct comparison between the two time intervals. But, there are some cases where time constraint on the explicit transitions is not presented explicitly. For example in Figure 3-3, we can extract that the time constraint on the **enable-disable(-)** message between the state *s3* and *s5* is ($X1 \in [0,1[\vee X3 \in [0,1[$). This constraint is not explicitly presented on the transition but it is extracted from the constraint on the implicit transition. Therefore, there are two ways to deal with implicit transitions during the compatibility and replaceability analysis. The first way is to use well known modeling tools such as the Timed Automata (Alur and Dill [5]) and the second way is to remove these implicit transitions. In the second solution, the protocol is reconstructed with only explicit transitions annotated explicitly with the time constraints which present the semantics of the removed implicit transitions.

Timed Automata are simple, powerful, and a widely known formalism. Therefore, it can be used to capture a large class of Web services by implementing the business protocols of these services. The main problem of using timed automata to implement business protocols of Web services is implicit transitions. This is because this type of transitions has no equivalent transitions in the context of the timed automata with the same semantics. The semantics of the epsilon transition on the timed automata (silent transition) is different from the semantics of the implicit transition on the business protocol. Ponge et al. [114] use the epsilon transition of timed automata for representing the implicit transition and modify its semantics with some additional time constraints. They convert the business protocol to timed protocol automata to be used in the compatibility checking analysis based on the

language inclusion between the two protocols. For instance, if the conversations that are presented by one protocol can be understood from the other protocol, then they are fully compatible. This approach cannot be used in our analysis, for two reasons. First, our definitions of compatibility and replaceability checking are based on the error free interaction which is different from the previous definition which is based on the language inclusion (as shown in example 3.1). Second, their conversion technique does not work with a wide range of Web services business protocols. For example, the protocol which has a loop and one of the transitions of the loop is an implicit transition with time constraint. Furthermore, the implicit transition on the loop has a clock reset and the time constraint on the implicit transition checks this clock (e.g., the implicit transition on Figure 3-3 has a clock reset

x3 and this clock is checked in its time constraint($x1=1 \vee x3=1$)).

The other way of dealing with the implicit transition is to remove it and preserve the semantics of the protocol. The idea is to replace this transition, which is based on time constraints related to explicit transitions, by time constraints reflecting its effect on the semantics of the protocol on the explicit transitions. Figure 3-3 and Figure 3-4 show two business protocols in which the first one has an implicit transition and the other not. As shown in example 3.1, compatibility checking in terms of time constraints is based on the direct comparison between the two time constraints on each transition. But due to the implicit transition, there are some situations where the time constraint on each explicit transition is not explicitly annotated. Therefore, during the checking process, for each explicit transition, we have to calculate the time constraints which are deduced from the presence of implicit transitions. As a result, the compatibility checking between the two protocols, in Figure 3-3 and Figure 3-4, in this form is a very hard task because of the overhead calculation of time constraint for each explicit transition and the different representations of the two protocols. The difficulty in this example can also come from the constraint on the implicit transition which checks the implicit clock *x3*. Furthermore, checking the compatibility between the two protocols without the conversion of the first protocol and using direct comparison for the time constraints, without calculating the effect of the implicit transition on the explicit ones, shows

that the two protocols are not compatible (which is a false result). But after transforming the first protocol into a new one without implicit transitions, the checking result shows that the two protocols are compatible (which is the true result).

Timed Web services can have one clock or more depends on the function of the service. In the one clock Web services, there is only one clock used by the service and this clock is reset when a transition is triggered. But, multi-clocks Web services use a set of clocks and each clock is associated with one of the transitions and reset when this transition is triggered. Thus, the conversion process is a complex task and depends on the following parameters

- The Web service uses one clock or multi-clocks
- The implicit transition constraints,
- The clock resets on the implicit transition
- The implicit transition is included in a loop or not.

For timed Web services that use one clock, the conversion process is not hard. But in the multi-clocks timed Web service, the simplest conversion examples are those for which the implicit transitions do not have a clock reset and the hardest examples are those for which implicit transitions have an implicit clock, a time constraint using this clock, and these implicit transitions are included in loops.

Therefore, we first present the timed Web services business protocol that uses one clock. After that, we present the multi-clock business protocols with a general approach for removing any form of implicit transition without changing the semantics of the protocol. This solution deals with multi-clock protocols where the clocks on the time constraints on any transitions are not necessary the clocks of the previous transitions but any clock for any transition. After removing the implicit transitions, compatibility and replaceability algorithms can work in a straightforward way.

3.4 One clock timed business protocols

The time constraints on the Web services that use one clock restrict the sending or a message to specific time interval values of this clock. These time intervals contain the possible values of the clock that permit the service to send or receive messages. This clock is reset each time the transition triggers. As a result, the value of the clock at any time present the time passed since the triggering of last transition.

The problem of the type of one clock Web service is that the time constraints are only depending on the last previous triggered transition. The definition of the timed business protocol is based on the definition of Benatallah et al. [19]. This protocol is deterministic (i.e., all the output transitions from any state are different and there is no overlapping between the output messages).

Definition 3-1. A One-clock timed business protocol with implicit transitions *(OCTBP) is a 6-tuple* $P = (S;s_0;x_g;T;T_i;F)$ *which consists of the following elements:*

- S *is a finite set of states.*

- $s_0 \in S$, *is the initial state.*

- x_g *is a global clock.*

- $T \subseteq S^2 \times M \times \{+,-\} \times Tc$, *is a finite set of explicit transitions where M is a set of input/output messages,* $\{+,-\}$ *is the polarity of the message where* $\{+\}$ *means input message and* $\{-\}$ *means output message. Tc is the time constraint which is in the form* $Tc \subseteq (x_g \times \Phi \times a)$, *or* $Tc \subseteq (x_g \in I)$, *where* $\Phi \in \{<;>;<=;>=;=\}$, *I is the set of time intervals in the form* $I = [a,b[$ *or* $I =]a,b]$, *and* $a,b \in \mathbb{R} \cup \{\infty\}$.

- *$T_i \subseteq S^2 \times Tc$, is a set of implicit transitions with time constraints tc $\{x_g=a\}$.*

- *$F \subseteq S$ is the set of final states. If $F = \emptyset$ then P is said to be an empty protocol.*

- *All states in the protocol are accessible and co-accessible.*

The timed business protocol is represented as a state chart which consists of a set of states containing an initial state and one or more of final states and a 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 could be assigned with time constraints and clock reset only, and the explicit transition could be assigned with messages, time constraints and clock reset. 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

An example of one clock Web service business protocol is shown in Figure 3-5. The clock x_g is the clock used by the service which is reset with the triggering of each transition. The transition between the states *S2* and *S3* is an implicit transition with time constraint $x_g=a$, where *a* is a time value. Based on the semantics of this time constraint, after the triggering of the transition between the states *S1* and *S2*, the transition between the states S*2* and *S4* can be triggered in the time period [0,*a*[. Otherwise, the transition between the states *S2* and *S3* will be triggered and by consequence the transition between the states *S3* and *S4* can be triggered. It is clear in this example that the value of the checked clock is based on the triggering of the transition between the states *S1* and *S2*.

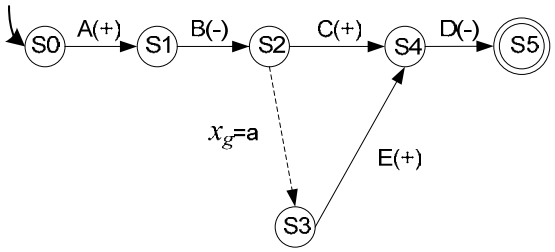

Figure 3-5: One clock Web service business protocol with implicit transition.

In order to ease the analysis of such protocols, we perform the conversion of implicit transitions to time constraints on explicit transitions. The new business protocol is called **one clock explicitly time business protocol**.

Definition 3-2. A one clock explicitly time business protocol (OCETBP) *is a 5-tuple P = (S; s_0; xg; T; F) which consists of the following elements:*

S is a finite set of states.

$s_0 \in S$, is the initial state.

x_g is a global clock.

$T \subseteq S^2 \times M \times \{+, -\} \times Tc$, is a finite set of explicit transition where M is set of input/output messages, $\{+, -\}$ polarity of the message where $\{+\}$ means input message and $\{\}$ means output message. Tc is the time constraint which is in the

form ($x_g \in I$) where I is the set of time intervals in the form I =[a,b[or I=]a,b],

and a,b $\in \mathbb{R} \cup \{\infty\}$..

- *This protocol is deterministic (i.e., for each state, all the partial paths from any state are unique).*

- $F \subseteq S$ *is a set of final states. If* $F = \emptyset$ *then P is said to be an empty protocol.*

- *All states in the protocol are accessible and co-accessible*

Figure 3-6 shows an example of one clock Web service business protocol **without** implicit transitions. This semantics of this protocol is the same as the sementics of the protocol of Figure 3-5.

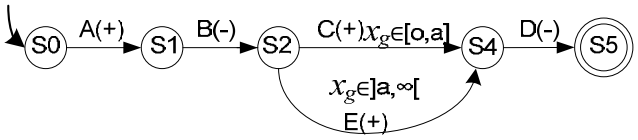

Figure 3-6. One clock Web service business protocol **without** implicit transition

We can perform this conversion using algorithm 3-1**.**

Algorithm 3-1: Conversion of timed business protocol *P* with implicit transition to explicitly business protocol.

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

$I_k = [0,t[\cap I_k$

Else

forall $(s_i; s_k; m_k^{\pm} ; I_k) \in T_e$ *where* $s_k \in S$, $m_k \in M$,$0 \leq k \leq n$, n *is number of states in protocol P* ***do***

$I_k = [0;\infty [\cap I_k$

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

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

foreach $(s_0; s''; \overrightarrow{m}; I(x; y)) \in T_e$ ***do***

$T_e = T_e \cup (s_0; s''; \overrightarrow{m}; I(x + t; y + t))$

Delete transition IT(s,s',t)

Return *P*

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

Figure 3-7 shows an example of a business protocol with implicit transition and Figure 3-8 shows its equivalent business protocol without implicit transition. We explain the idea of the algorithm using these two protocols. It can be noticed from the time business protocol with implicit transitions that the implicit transitions effect on the time constraints of two types of explicit transitions.

1- The explicit transitions which gets out from the same state of this implicit transition. For example, in Figure 3-7 the implicit transition between the states S_1 and S_3 has an effect on the explicit transition between the states S_1 and S_4. As a result, if we delete this implicit transition we have to add his effect on the explicit transition, in this case the triggering of the transition between the states (S_1 and S_4) is restricted by the time constraint $x_g \in [0, t_1[$. The same process will be done for the implicit transition between the states S_4 and S_5 with the time constraint $x_g = t_3$ which affects on the transition between the states S_4 and S_6 and the time constraint will be $x_g \in [0, t_3[$.

2- Any explicit transition has an implicit path before its source states. For example, in Figure 3-7:

A) The time constraint $x_g=t_3$ on the implicit transition between the states S_4 and S_5 has an effect on the transition between the states S_5 and S_7 because it restricts its triggering to the time constraint $x_g \in [t_3,\infty[$ if the protocol takes the path (S_1 ,S_4,S_5,S_7) (i.e., there is an explicit transition between S_4 and S_7 with time constraint $x_g \in [t_3,\infty[$).

B) The constraint $x_g=t_2$ on the implicit transition between the states S_3 and S_4 has an effect on the transition between the states S_4 and S_6 because it restrict its triggering to the time constraint $x_g \in [t_2,t_2+t_3[$ if the protocol takes the path (S_2 ,S_3,S_4,S_6) (i.e., there is an explicit transition between S_3 and S_6 with time constraint $x_g \in [t_2, t_2 + t_3[$).

C) The constraint $x_g=t_1$ on the implicit transition between the states S_1 and S_3 and the constraint $x_g=t_2$ on the transition between the states S_3 and S_4 have an effect on the transition between the states S_4 and S_6 because they restrict its triggering to the time constraint $x_g \in [t_1+t_2, t_1+t_2 + t_3[$ if

the protocol takes the path (S_1,S_3,S_4,S_6) (i.e., there is an explicit transition between S_1 and S_6 with time constraint $x_g \in [t_1+t_2, t_1+t_2+t_3[$).

D) The constraint $x_g=t_2$ on the implicit transition between the states S_3 and S_4 and the constraint $x_g=t_3$ on the transition between the states S_3 and S_4 have an effect on the transition between the states S_4 and S_6 because they restrict its triggering to the time constraint $x_g \in [t_2+ t_3,\infty [$ if the protocol takes the path (S_2,S_3,S_4,S_5,S_7) (i.e., there is an explicit transition between S_3 and S_7 with time constraint $x_g \in [t_2+ t_3,\infty [$).

E) The constraint $x_g=t_1$ on the implicit transition between the states S_1 and S_3 and the constraint $x_g=t_2$ on the transition between the states S_4 and S_5 have an effect on the transition between the states S_5 and S_7 because they restrict its triggering to the time constraint $x_g \in [t_1+t_2+ t_3, \infty [$ if the protocol takes the path (S_1,S_3,S_4,S_5,S_7) (i.e., there is an explicit transition between S_1 and S_7 with time constraint $x_g \in [t_1+t_2+ t_3, \infty [$).

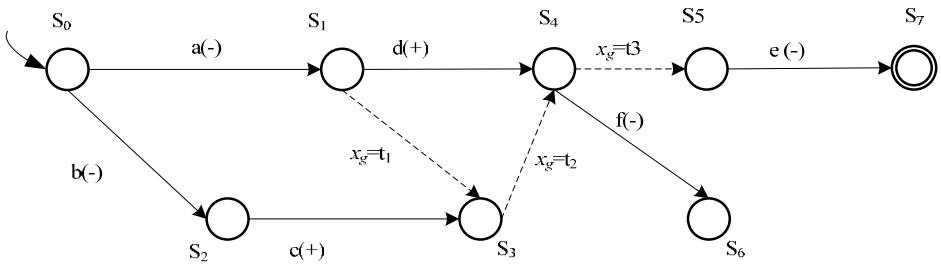

Figure 3-7: Business protocol of Web service with implicitly transition.

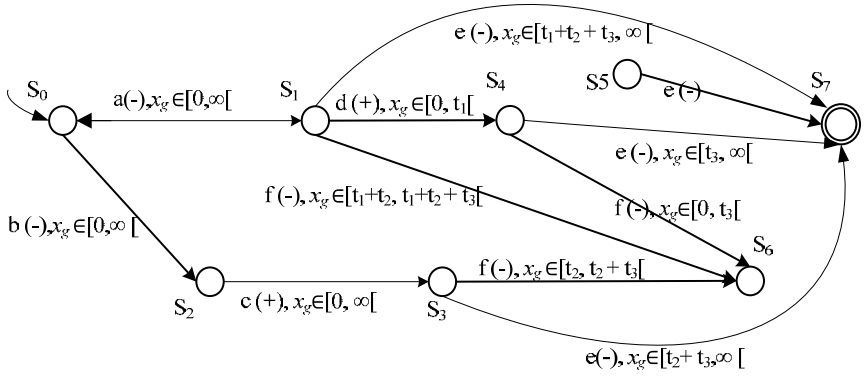

Figure 3-8: Business protocol of a Web service without implicit transitions.

3.5 Multi-clocks timed protocol modeling

Many times Web services rely on multi-clocks for presenting their temporal constraints. Simply, multi-clocks services could assign to any transition a clock that is reset when this transition is triggered and checked in any time during the interaction with the Web service. This section presents the formal definitions of multi-clocks timed business protocols and the approaches that are used for the automated analysis. We will start by defining the static atomic time constraint on the timed business protocols.

Definition 3-3 ***(Static atomic time constraint)*** *A static atomic time constraint on a timed business protocol (TBP) is either* $x \in \#a,b\#$ *or* $x=a$ *where:*

- $x \in X$ *and* X *is a finite set of clocks.*

- $a, b \in \mathbb{R}$ *are relative real time points.*

- $\# \in \{[,]\}$

The static atomic time constraint is the simplest form of time constraints that can be annotated on the timed business protocol to restrict the triggering of the transitions to specific time windows. This constraint can take two formats; the first format is the time interval format which explicitly presents the time constraint in terms of time interval which restricts the triggering of the transition to this time interval. For example, $x1 \in [1,5[$ restricts the triggering of the transition to time values from one to five. The second format is the equality format which restricts the triggering to a fixed time point. For example, the constraint ($x1$=5) restricts the triggering of the transition to time value 5.

Definition 3-4 ***(Multi-clocks timed business protocol) (MCTBP)*** *multi-clocks timed business protocol (MCTBP) is a 6-tuple* $P = (S;s_0;X;T;T_i;F)$ *which consists of the following elements:*

- *S is a finite set of states*

□ $s_0 \in S$, *is the initial state.*

□*X is a finite set of clocks with a set of clock reset* $CR \in X$.

□$T \subseteq S^2 \times M \times \{+, -\} \times Tc \times CR$, *is a finite set of explicit transitions where M is a set of input/output messages,* $\{+, -\}$ *is the polarity of the message where {+} means input message and {□} means output message. Tc is the time constraint which is a disjunction of the conjunction of static atomic time constraints.*

□$T_i \subseteq S^2 \times Tc \times CR$, *is a set of implicit transitions with time constraints having at least one equality for each disjunction.*

□$F \subseteq S$ *is the set of final states. If* $F = \emptyset$ *then P is said to be an empty protocol.*

□*All states in the automata are accessible and co-accessible.*

The static atomic time constraint consists of the clocks names and the values for these clocks that determined the allowed time interval for the triggering of the transition. This time interval is fixed time interval in each time of the transition trigger-

ing (i.e., each time the transition triggers it will obey to the fixed time values on the interval). Therefore, we called this constraint static time constraint. There are some cases where this time constraint interval is not static. This means that each time the transition is triggered based on a different time constrains interval for the same clock. We called this type of time constraint "dynamic time constraint". In the dynamic time constraint the values on the time constraint intervals are presented by variables in addition to constants. The transition that has a dynamic time constraint is triggered using a different value of the variable in each triggering.

Definition 3-5 ***(Dynamic atomic time constraint)*** *A dynamic atomic time constraint on a transition (t) is DAC* $\subseteq X \times \{\in\} \times [k*c+d; k*c+d[$ *where c, c', d, d' are constants , and k is a variable obeying to the sequence of natural numbers.*

An example for a dynamic atomic time constraint is the time constraint ($x1 \in [0,k[$).

This time constraint means that in the first triggering of the transition the variable k will be set to one (i.e., k=1) and the constraint becomes ($x1 \in [0,1[$) and in the second triggering the value of k is equal two and the constraint becomes ($x1 \in [0,2[$) and in the n^{th} triggering the constraint becomes ($x1 \in [0,n[$).

Definition 3-6 ***(Multi-clocks explicitly time business protocol) (MCETBP)*** *An explicitly time business protocol is a 5-tuple P = (S; s_0; X; T; F) which consists of the following elements:*

□*S is a finite set of states.*

□*$s_0 \in S$, is the initial state.*

□*X is a finite set of clocks with a set of clock reset $CR \in X$.*

□ *$T \subseteq S^2 \times M \times \{+, -\} \times Tc \times CR \times RK$, is a finite set of explicit transition*

where M is set of input/output messages, $\{+, -\}$ polarity of the message where {+} means input message and {□} means output message. Tc is the time constraint which is a disjunction of the conjunction of static and dynamic atomic time constraints. RK is the set of variables to reset, if there is no dynamic variable in the protocol then this set is always empty.

□ *This protocol is deterministic (i.e., for each state, all the partial paths from any state are unique).*

□*$F \subseteq S$ is a set of final states. If $F = \emptyset$ then P is said to be an empty protocol.*

□*All states in the protocol are accessible and co-accessible.*

The previous definition shows that the explicitly timed business protocols do not have implicit transition and the time constraint can contain dynamic time constraints. The *RK* set on each transition contains the variables that are used in the dynamic constraints and reset when this transition is triggered.

3.5.1 Timed business protocol semantics

A timed business protocol expresses two types of constraints on the external behavior of a given service. The first type is the conversations that a service supports which is expressed in terms of sequences of message exchanges. This conversation constraint can be characterized using the so-called linear time process semantics in which a process is completely determined from the set of its (partial) observable runs (or traces). By using this approach, the behavior of a protocol will be characterized in terms of all its observable traces. For example, the sequence of message exchange ***Opensession(+).IDrequest(-).IDresponse(+). Enable-disable(-)*** is allowed by the protocol given in Figure 3-3.

The second type is the timing constraint which is specified when a given message is enabled to occur inside a conversation. This constraint is characterized by extending the concept of trace to include timing constraints. A correct execution (or simply execution) of timed protocol P is a sequence $\sigma=s_0.(m_0,t_0).s_1 \ldots s_{n-1}.(m_{n-1},t_{n-1}).s_n$ such that: (i)$t_0 \leq t_1 \ldots \leq t_n$ (ii) s_0 is the initial state and s_n is a final state of protocol P, and (iii) $\forall j \in [1,n]$, we have $(s_{j-1},s_j,m_{j-1},tc_{j-1})$ where the clock values at the time t_{j-1} satisfy the time constraint tc_{j-1}. As an example, the sequence σ= ***start.(login(+),0). Logged. (ReqCard(-),4). CardWait.(ε,54). Failure ACK.Access denied(-),60).Fail*** is a correct execution of the protocol depicted at Figure 3-9. If the execution σ is a correct execution of protocol P, then the time trace $tr(\sigma)$ is compliant with P. Continuing with the example, the execution σ leads to the time trace ***tr(σ)=(login(+),0).(ReqCard(-),4).(ε,54).(Accessdenied(-),60).*** The timed conversation of a protocol P is obtained by removing from the corresponding time trace $tr(\sigma)$ all the non observable events (i.e., all the pairs (m_i,t_i) with $m_i=\varepsilon$). For example, the timed conversation of the execution σ is

con(σ□)=(login(+),0).(ReqCard(-),4).(Accessdenied(-),60). Another example of time conversation is the sequence of timed message exchange: **(*Opensession(+), 0),(IDrequest(-),1.5), (IDrespose (+),1.8) ,(Enable-Disable(-),1.9)*** which can be recognized by the protocol of Figure **3-3**. Each term of this timed conversation consists of the message and its polarity, and the time instance in which the message occurs. We denote *Tr*(*P*) the, possibly infinite, set of timed conversation of *P*. Two protocol *P1* and *P2* are semantically equivalent if *Tr(P1)=Tr(P2)*. If there is a clock reset when the message is sent or received, then the Clock Reset Value (*CRV*) of this clock is equal to the time instance when the transition which has this message is triggered. For example in Figure 3-3, the transition of the ***Opensession(+)*** message has a clock reset *x1* and the *CRV*(*x1*)=0. The function *CRV*(*x*) determined time in which the clock *x* is reset during the interaction with the protocol. For example, *CRV* (*x*)=*t*□ iff clock *x* has been reset at time *t*□after the start of the interaction with the protocol. After resetting a clock *x* at time t (i.e., *CRV(x)=t*), the value of this clock at any time *t*□after this resetting is equal to the difference between the time value *t*□and the time value *t* represented by the *CRV(x)*. This difference which presents the clock value can be calculated by the Clock Value

function $CV_t(x):X \rightarrow \mathbb{R}$, with $x \mapsto t\text{-}CRV(x)$ that maps the set *X* into the set *R* by

mapping each clock *x* to its value *t*-CRV*(x)* that belongs to $\mathbb{R}$. The satisfaction of

the time constraint *tc* with a clock value, which is calculated by $CV_t(x):$, is referred to by $[tc]_{CV\,t}(x):$. The value of $[tc]_{CVt}(x):$is true if the value of each clock *x* at the time *t* satisfy the time constraint *tc*. **Output (s_i)** defines all the outgoing transitions triggered from the state (s_i) and an **Input(s_i)** defines all the incoming transitions to the state (s_i).

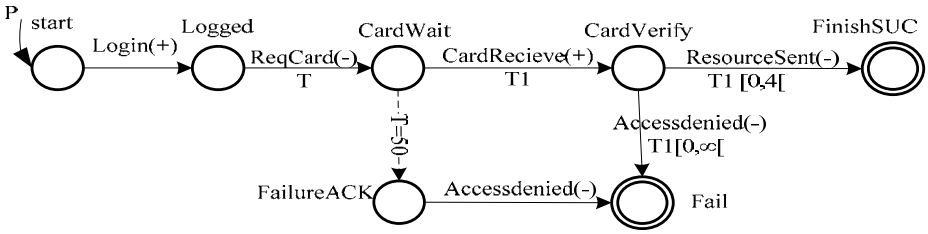

Figure 3-9: A Web service business protocol *P* with implicit and explicit transitions.

Definition 3-7 ***(Implicit path)*** *An implicit path IP (s,t,s□,t□) in a TBP (S; s_0; X; T; T_i; F) where s, s□∈S, and t, t□∈R, is*

Either *an empty path (i.e., s=s□and t=t□),*

Or *there exist s□, t□such that (s, t, s□, t□) is an implicit path in TBP and there exist TC□and CR□such that*

- *(s□,s□,TC□,CR□) ∈ T_i, t□≤=t□, there is no (s□,s□□,TC□□,CR□□) ∈ T_i, and no t□≤=t□□≤t□, such that* $[tc']_{cv(c')}(x)$ *=true.*
- $[tc']_{cv(c')}(x)$ *= true, x ↦ t□if x ∈*CR.

The implicit path consists of one or more consecutive implicit transition. Because of the deterministic property of the business protocol, there is no more than one implicit transition as an output transition in any state of the protocol. Therefore, this property is inherited by all paths in the protocols which include implicit paths.

Definition 3-8 ***(Time conversation of TBP)*** *A time conversation trace* $(m_i,t_i)_{k\le j\le k}$ *is recognized by TBP (S; s₀; X; T; Tᵢ; F) from s,t if*

Either

- *There exist an implicit path* $(s,t,,s',t')$ *where* $t\le=t_i$ *in TBP*

- $\exists s'', tc'',CR''$ *such that* $(s',s'',m_i, tc'',CR'') \in T$ *, where*

$$[[tc']]_{(}[[CV(t)]]_{)}(x) = true\ ,\ x \mapsto t_i \text{ if } x \in CR''$$

- *There is no implicit path* (s,t,s''',t''') *in TBP such that* $t\le t'''\le t$.
- $((m_j,t_j)_{k+1\le j\le k}$ *is recognized by TBP from* s'', t_i .

Or

$k\le k$ *(empty trace) and there exist an implicit path* (s,t,CRV,s',t', CRV') *in TBP such that* $s\in F$.

- *The time conversation is complete if it starts by the state* s_0 *and ends by* $s\in F$

Definition 3-9 (Time conversation of ETBP) *A time conversation trace* $((m_i,t_i)_{l\le j\le l}$ *is recognized by ETBP (S; s₀; X; T; Tᵢ; F) if*

- *There exist* s', tc', CR' *such that* $(s_i, s', m_i, tc', CR') \in T$

- $(tc')_{|CR'}(x) = true,\ x \mapsto t_i$ *if* $x \in CR'$

- $((m_j, t_j))_{l+1 \leq j \leq l'}$ *is recognized by ETBP from state* s_j.
- *The time conversation is complete if it starts by the state* s_0 *and ends by the state* $s \in F$.

A timed business protocol without implicit transitions (MCETBP) presents two types of constraints, the sequence of message exchange and the time constraints. The time constraints in the MCETBP and in MCTBP are presented in a different manner. Two types of time interval constraints will be presented on the transition. The first is the static time interval where the clock value is checked by a fixed time interval with two constant limits (e.g., the time constraint $x1 \in$ [1,7[checks if the value of the clock *x1* is between the two values one and seven). The second type of time constraints is the dynamic interval time constraint where at each triggering of the transition, the time interval constraint is different. The time constraint $X1 \in$ *{[a,∞[, [a+b,∞[,]a+2*b,∞[,□.} with a=5,b=2* on the transition between the

states *S* and *S*□on Figure 3-10 is an example of the dynamic interval time constraint. Based on this constraint, the time interval constraint on the transition on the

first triggering is *X1*∈ [5,∞[, in the second triggering is *X1*∈ [7,∞[, and so on. This

constraint can be rewritten in another way *x1*∈*[a+k*b,∞[* with *a=5,b=2,*and

k={0,1,2,□*}* where *k*+1 represent the triggering number (*i.e., k=0* is used in the first triggering). Therefore, the clock *x1* will be checked against different time intervals based on the sequences of value of the variable *k* in each triggering.

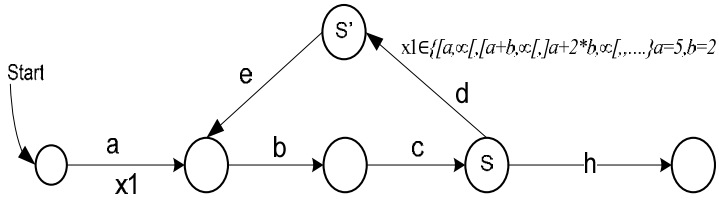

Figure 3-10: Business protocol contains dynamic interval time constraint on one of its transitions.

3.6 Related work

Modeling Web services behavior for automated analysis has gaining a great importance. Based on the functionality of the Web service, it may contain a time constraints or not. For modeling Web service behavior without time aspects one can refer to Beyer et al. [32], Bultan et al. [41], Bultan et. al. [40, 42], Fu et al. [71, 72], Honda et al. [81], and Gay et al. [73].

Beyer et al. [32] define protocol interface formalism for services which is similar to the timed model proposed by Berardi et al. [23] but without time aspects. A language for specifying Web service interface is presented. They specified three kinds of constraints by the interface on the users of the service; signature constraints,

consistency constraints, and protocol constraints. In signature constraints, the interface specifies the methods that can be called by a client, together with types of input and output parameters. In consistency constraints, the interface may specify propositional constraints on method calls and output values that may occur in a Web service conversation. In protocol constraints, the interface may specify chronological constraints on the ordering of method calls. They use these interfaces in checking compatibility and replaceability (substitutability). This work presents protocol interface formalism supporting programming and modeling language constructs which are supported by Web service programming or modeling frameworks like the .NET framework, or Web Service Choreography Interface (WSCI).

Bultan et al. [41] present a modeling for Web services interaction by specifying the global behavior of e-services compositions. They present formalism for specification and verification of electronic services for composition purposes. They extend this work by studying the realizability problem [72]. Bultan et. al. [40, 42] present three modeling formalism for interactions among Web software: 1) Collaboration diagrams, 2) Message sequence charts, and 3) Conversation protocols and realizability and synchronizability problems. This work present modeling formalism but does not define any compatibility and replaceability checking.

Analyzing and verifying properties of composite Web services specified as multiple BPEL processes are performed by X. Fu et al. [71]. The services are specified using automaton based formalism and the properties that needed to be checked are expresses using temporal logic (Pnueli [112]). So, they present an approach that analyzes some given properties in a given composition, whilst we are interested in the compatibility analysis that can be used to build a composition.

Session types can be effectively used for describing protocols as shown by Honda et al. [81] and Gay et al. [73]. Vallecillo et al. [132] use session types and theory of subtyping (Gay and Hole [74]) to formalize compatibility and substitutability of components. Carbone et al. [44] and Mostrous et al. [102] present formalism, called global calculus, aims at representing global message flows as structured communications. Session types offer a high-level abstraction and articulation for complex communication behaviors presents Web service business protocols. They show how session types allow the definition of powerful interoperability tests at the protocol level such as compatibility and substitutability. However, time aspect is not presented in all the previous work on session types.

Some works concentrate on time as an effective player on systems behaviors. For instance, Maria et al. [97], Tiplea and Macovei [130], and Bettini et al. [31] studied time as an important parameter in workflow systems. Maria et al. [97] discussed the modeling and verification of workflows extended with time constraints. They used the timed automata as an effective tool to specify workflow schemas with time constraints and to check their consistency. Tiplea and Macovei [130] studied the time workflow nets where a time interval is associated to each transition. Bettini et al. [31] model advances temporal features for workflow. They enhance the capabilities of the workflow systems by specifying and reasoning on temporal constraints about the duration and distance among activities that compose these systems. These constraints restrict the durations of activities using an execution deadline or a time window within which the activity must be performed. Furthermore, workflow system participants can get information by reasoning about the time and the allowed period in which they may engage in an activity due to the overall temporal constraints.

Berardi et al. [23], Dyaz et al. [62], and Kazhamiakin et al. [88] studied time in Web services. Timing issues can be involved in many types of protocols that represent systems behaviors such as business protocols [19]. Time-related behaviors can be session timeouts or deadlines with different kinds of behaviors (e.g., visa card must be provided within *n* hours, otherwise the service will be cancelled). However, usual model checking verifications such as liveness, safety, and testing the absence of deadlocks are performed by most of the approaches which consider time [23, 62, 88].

Luca de Alfaro et al. [56, 55, 46] have presented rich interface formalisms for software components whose correct interaction depends on timing, communication protocols, or resource use. They presented timed interfaces and checked the compatibility between them for composition. In their game-based approach, input and output play dual roles: two components are compatible if there is *some* input behavior such that, for *all* output behaviors, no incompatibility arises. This game-based approach is based on software component with explicit transitions (action) only and cannot handle implicit transitions based on time-out, which is the case in Web service protocols. Also, they use one clock for each component in their formalization which is not always the case in Web services.

Berardi et al. [23] propose a language for representing e-services (electronic services) behavior namely Web Service Transition Language (WSTL), taking time

constraints into account. This language integrates with standard language like WSDL in order to specify e-services. The e-services behavior is represented as finite state automata. They present time out constraints for input and output messages. Related to our work, they only present the time constraint concept on the description of the behavior of e-service. But, they did not make any analysis in terms of compatibility or replaceability.

Some work that considers timing abstractions has been done by Kazhamiakin et al. [88]. They mainly reuse well-known timed automata model-checking techniques in service-based compositions. However, this model does not consider data flow, external constraints and constraints over counters and more generally constraints over data. They present straightforward checking for Web service properties. Moreover, this work does not consider the compatibility and replaceability analysis.

In the same line with the work presented by Fu et al. [71], timed automata are used to check the timed properties of a given composition by Dïaz et al. [54]. They translate the descriptions written in WSCI-WSCDL [87] into timed automata. They deal with the problem of verifying a given composition, whilst we are interested in the compatibility analysis that can be used to build a composition.

Benatallah et al. [16] present an algorithm for calculating the time windows for each explicit transition before checking the compatibility. This algorithm works with simple business protocols which do not have implicit transition on loops with implicit clock reset or implicit clock on the constraint.

Ponge et al. [114] handle temporal aspects and built their analysis on the definition of compatibility (full compatibility and partial compatibility) given in [19, 20]. They converted the timed business protocol to a new class of timed automata called protocol timed automata and defined a set of operators, such as composition, intersection, and projection. For instance, to check full compatibility between protocols *P1* and *P2*, they check if the projection by *P1* of the composition between *P1* and *P2* is equal to *P1* (i.e., if ([*P1* COMPOSITION *P2*] PROJECTION *P1*) = *P1*, then *P1* is fully compatible with *P2*). Guermouche et al. [78] studied the compatibility checking by considering operations, messages, data associated with messages, and conditions on data and temporal constraints.

The works reported in [114] and [78] use compatibility checking definitions different from our definitions as explained in Section 3.2. Therefore, their way of model-

ing and analysis is not suitable with our approach because we have different definitions based on transition-transition comparison.

To the best of our knowledge, all the mentioned works are not able to check compatibility and replaceability between business protocols which have the following properties using our compatibility and replaceability checking definition: 1) The implicit transition time constraints are complex 2) The time constraint on the implicit transition contains implicit clocks in it 3)The implicit transition has an implicit clock reset 4) The clock system of the protocol is based on the multi-clock system where each clock is attached with a clock and this clock is reset when this transition is triggered. Therefore, there is a need for removing implicit transitions which is a crucial step towards a complete compatibility and replaceability checking.

Chapter 4 .IMPLICIT TRANSITION REMOVAL APPROACH

This chapter presents the conversion approach of business protocols after removing the implicit transitions without changing the semantics of the business protocols. It is divided in two main parts; the first part shows the separation approach and the second part shows the main conversion approach [65].

4.1 Implicit transition constraint separation approach

The implicit transition is triggered when its time constraint is satisfied. The satisfaction of its time constraint is achieved by the arrival of one of the clocks on the constraint to a specified value. This value, which is in the equality condition (e.g., *x1=2*), is a triggering instance for this transition. The constraint on the implicit transition can have more than one of the triggering instance values (e.g., the time constraint *(x1=2 ∨ x2=3)* on an implicit transition means that this implicit transition can be triggered when the value of the clock *x1* reaches to two or *x2* reaches to three). The implicit transition separation approach is used in the cases where the implicit transition has more than one triggering instance value. This approach is needed because we do not compare directly two clocks. It is used for replacing the implicit transitions which are triggered by a time constraint in the form $Tc(ti) = \bigcup_{i=1}^{n} Tc(ti)_i$ (i.e., more than one atomic constraint separated by the logical operator OR , e.g., *TC={ x1=2 ∨ x2=5)* to an equivalent *n* implicit transitions where each

transition can be triggered by only one atomic constraint of *Tc(ti)*. The semantics of the business protocol will not be changed after the replacement process. The idea behind this approach is to get an equivalent protocol in which each implicit transition can only be triggered by one instance triggering value. In order to accomplish this task, we have to track the effect of the implicit transition and its exact semantics which can be substituted by the right time constraints. Figure 4-1 shows an example of two semantically equivalent business protocols (*P1* and *P1*□) and two semantically nonequivalent business protocols (*P2* and *P2*□). This example indicates the importance of the separation approach as an important step in the approach of removing the implicit transition. The difference between the protocols *P1* and *P2* is that the implicit transition constraint between the states *S1* and *S2* of the protocol *P1* checks only the clock *x1* (i.e., there is only one clock instance value that can triggers this implicit transition) but the implicit transition constraint on the protocol *P2* checks two clocks (*x1* and *x2*). On the protocol *P1*, the transition between the states *S2* and *S3* will be triggered only for all values of $x1 => 7$ and the transition between *S1* and *S5* will be triggered for all values of $x1 < 7$ if the value of *x1* is less than or equal 7 when the transition c triggers and for all values of $x1 > 7$ if the value of *x1* is greater than 7 when the transition *c* triggers. As a consequence , the effect of the implicit transition is to restrict the triggering of the transitions between the states *S2* and *S3* to $x1 \in [7,\infty[$ and between the states *S1* and *S5* to $x1 \in [0,7[$ *or* $x1 \in]7,\infty[$. Therefore, protocol *P1*□is semantically equivalent with protocol *P1*. On the protocol *P2*, the transition between the states *S2* and *S3* will be triggered for all values of $x1 => 7$ *or* $x2 => 5$ and the transition between *S1* and *S5* for all values of $x1 < 7$ *or* $x2 < 5$. As a consequence, the effect of the implicit transition is to restrict the triggering of the transitions between the states *S2* and *S3* to

$x1 \in [7,\infty[$ in some case and to $x2 \in [5,\infty[$ in some cases and between the states *S1* and *S5* to $x1 \in [0,7[$ in some cases and to $x2 \in [0,5[$ in some cases. But, we cannot delete the implicit transition on protocol *P2* and put the time constraint $x1 \in [7,\infty[\vee x2 \in [5,\infty[$ on the transition between *S1* and *S3* and the constraint $x1 \in [0,7[\vee x2 \in [0,5[$ on the transition between the states *S1* and *S5* as shown in the protocol *P2*□ because there are some time messages words that can be recognized by one of the two protocol and cannot be recognized by the other protocol. For example, the time word ***(d,0). (e,1). (f,3). (c,4). (h,6.5),*** can be recognized by the protocol *P2*□because the value of the clock *X1*, when the transition between *S1* and *S5* triggers , is equal six and five time unit which satisfy the constraint on the transition $(x1 \in [0,7[\vee x2 \in [0,5[)$ but this word cannot be recognized by the protocol *P2* because the clock *X2* reaches to the value five when the time reaches six (i.e., $x1=6$) and the implicit transition will be triggered and the transition between *S1* and *S5* will not be triggered after $t=6$ in this time word. Therefore, protocol *P2* and *P2*□are semantically nonequivalent and there is a need for a separation approach to make the implicit transition triggers based on one clock time instance value.

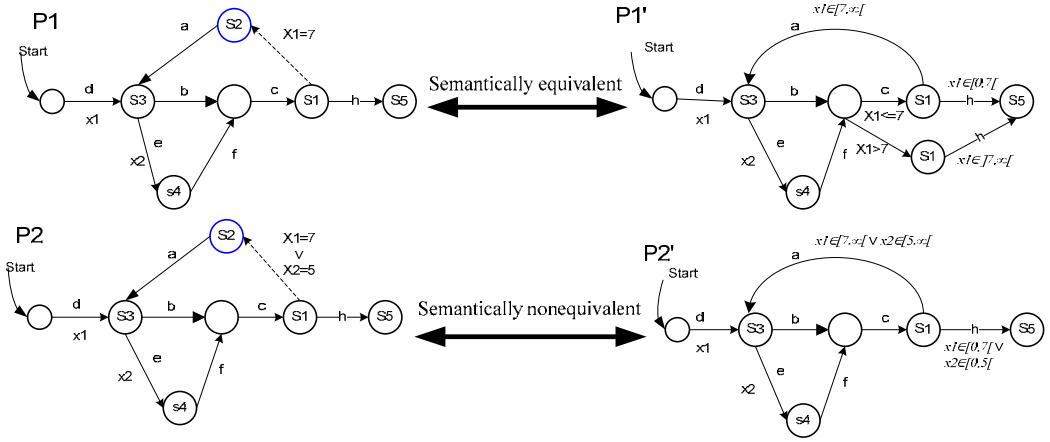

Figure 4-1: Two semantically equivalent protocols (*P1* and *P1*′) and two semantically nonequivalent protocols (*P2* and *P2*′).

Definition 4-1 (Path) *A path between two states s_i and s_j in a TBP is defined as*

$PA(s_i, s_j)=(s_i,m_i,s_{i+1},\ldots,m_{j-1},s_j)$ where $s_i,..,s_j \in S$ and $m_i,..,m_j \in M$. If $s_i=s_j$ (i.e., Path(s_i, s_j)=($s_1,m_1,s_2,\ldots s_n,m_n,s_1$)) then this path is in a loop form.

Definition 4-2 (Path-Loop-Path (PLP)) *A path-loop-path structure between two states s_i and s_j is $PLP(s_i, s_j)=(PA_1,L(s_1,s_2),PA_2)$ where PA_1 is a path starts by the state s_i and ends by the state s_1, $L(s_1,s_2)$ is a loop contains the states s_1,s_2 and PA_2 is a path starts by the state s_2 and ends by the state s_j.*

Example 4-1: Figure 4-2 shows an example of simple paths and loops and PLP. The path between the state s_0 and the state s_3 is $PA(s_0,s_3)=(s_0,a,s_1,b,s_2,c,s_3)$ is an example of a simple path and the path $(s_2,c,s_3,d,s_4,f,s_5,e,s_2)$ is an example of a loop. An example of a *PLP* is $(s_0,a,s_1,b,s_2,c,s_3,d,s_4,f,s_5,e,s_2,c,s_3,d,s_4,g,s_6,h,s_7)$ which is consists of the path (s_0,a,s_1,b,s_2), the loop $(s_2,c,s_3,d,s_4,f,s_5,e,s_2)$, and the path (s_4,g,s_6,h,s_7)

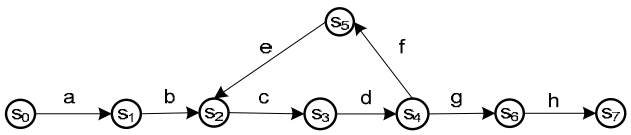

Figure 4-2: An example of simple paths, path in a form of a loop, and PLP form

The overlap between two paths is the shared path between the two paths. For example in Figure 4-3 (A), the path between the states s_0 and s_7, $PA(s_0,s_7)=(s_0,a,s_1,b,s_2,c,s_3,d,s_4,g,s_6,h,s_7)$ and the path between the states s_8 and s_7 , $PA(s_8,s_7)=(s_8,e,s_2,c,s_3,d,s_4,g,s_6,h,s_7)$ have an overlap $(s_2,c,s_3,d,s_4,g,s_6,h,s_7)$. This overlap can be a simple path or a loop or a PLP. We can separate this overlap without changing the semantic of the protocol (see Figure 4-3 (B)). An example of the overlap with PLP is shown in Figure 4-3 (C) and its equivalent part after the separation is shown in Figure 4-3 (D).

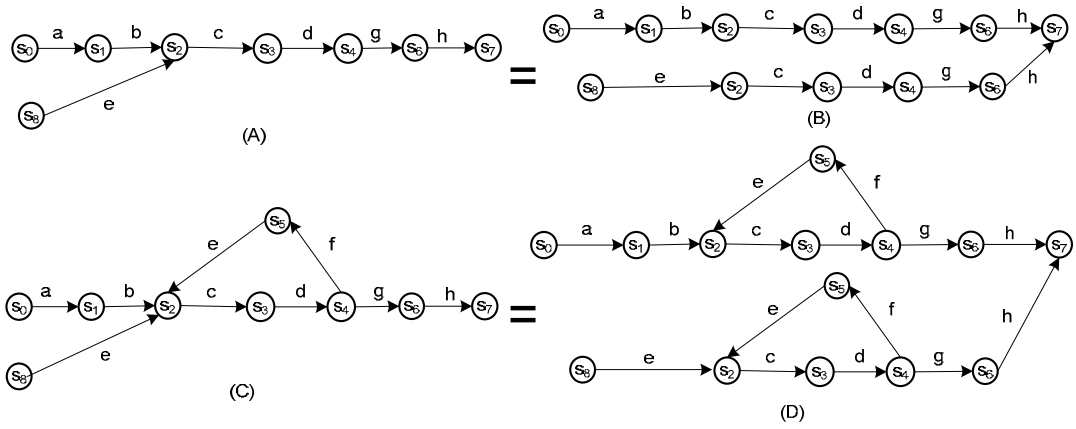

Figure 4-3: Examples of overlapped paths.

4.1.1 The separation approach

Input: Business protocol *P1* with implicit transition $t_i(ss_i,ds_i,tc_i,cr_i)$ where ss_i is the source state, ds_i is the destination state, $tc_i = (tc_{i1} \vee tc_{i2} \vee tc_{in})$ is the time constraint, and cr_i is the set of clocks reset when the transition t_i triggers.

Output: Business protocol *P2* semantically equivalent to the protocol *P1* with a set of *n* implicit transitions such that each implicit transition of the new *n* transitions is

in the form $t_{ij}(ss_{ij},ds_{ij},tc_{ij},cr_i)$ where $tc_i = (tc_{i1} \vee tc_{i2} \vee tc_{in})$ (i.e., each implicit transition of the new n transitions has a time constraint that can be satisfied by only one clock time instance value).

Begin:

For each implicit transition $t_i(ss_i,ds_i,tc_i,cr_i)$, where ss_i is the source state, ds_i is the destination state, $tc_i=\bigcup_{i=1}^{n} Tc(ti)_i$ is the time constraint, and cr_i is the set of clocks reset when the transition t_i triggers.

1. Create n copies of the implicit transition with new source states $(ss_i)_l$ and the same destination state ds_i. The explicit output transitions of each new state $(ss_i)_l$ are copies of the explicit output transitions of the state (ss_i) (See step one in Figure 4-4).
2. Assign to each implicit transition one of the disjunctive time constraints ($Tc(ti)_i$) of tc_i and remove the implicit transition t_i (See step two in Figure 4-4).
3. Create a new state (sa) with a copy of the explicit output transitions of the state ss_i of the implicit transition t_i. The inputs transitions of each state $(ss_i)_l$ and the state (sa) will be created in the next steps of the approach. The state $(ss_i)_l$ is the source for the implicit transition $(t_i)_l$ with time constraint $Tc(ti)_i$. Figure 4-4 shows an example of a part of a protocol and the form of this part after applying the three previous steps.

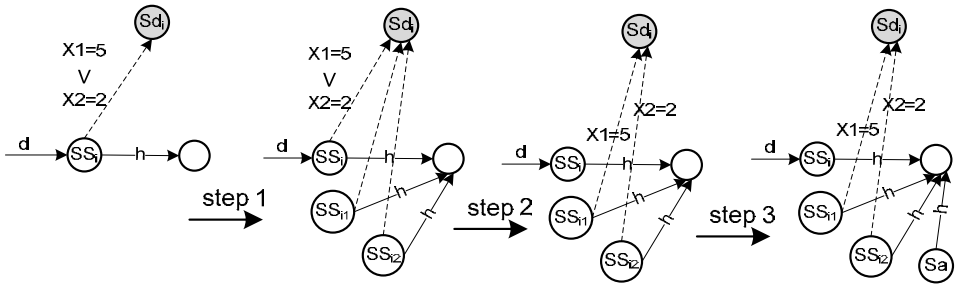

Figure 4-4: A part of business protocol has an implicit transition to show the effect of applying the first three steps of the separation approach.

4. Let *CK* be the set of clocks that will be checked, with equality operator, on the implicit transition t_i.
5. For all paths starting from the start state s_0 or the state ds_i to the state ss_i (i.e., $PA(s_0,ss_i)$ and $PA(ds_i,ss_i)$) .
   A. Arrange these paths by separating the overlapped paths. Figure 4-5 provides an example of business protocol *P1* and its equivalent protocol *P2* after the execution of this step. The path enters the loop through the state (s_{il}) and exits through the state (s_{ol}).

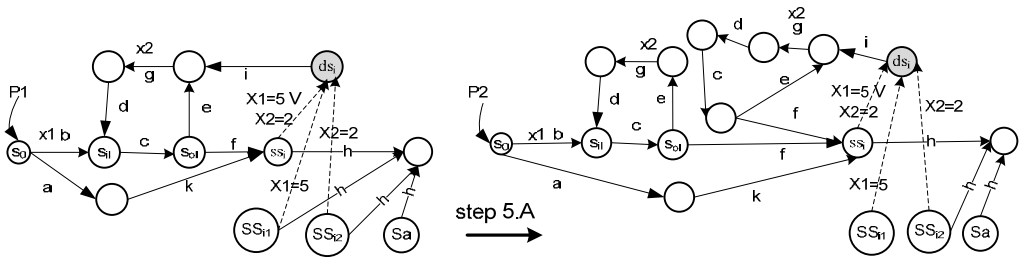

Figure 4-5: Business protocol P1 and its equivalent protocol P2 after the execution of step 5.A.

   B. For each state on the protocol, create an empty sequence called (Clocks Reset Sequence) $CRS(s)=\{\ \}$. The elements that will populate the sequence are the ordered clocks that are reset before this state and members of the *CK* set. The ordering of the clocks on the sequence is based on the time remaining for each clock to reach its constraint limit value on the implicit transition constraint tc_i.

   For example, if $tc_i = \{x1=v1 \vee x2= v2\}$ where $v1,v2 \in \mathbb{R}$ and $CRS(s)=\{x1,x2\}$,

   then this means that for all values of the clocks *x1* and *x2*, the values *x1-v1* are always greater than the values *x2-v2* in the state *s* (e.g., if the time constraint on the implicit transition is $tc_i = \{x1=5 \vee x2=2\}$ and the sequence on the

state s is $CRS(s)=\{x1,x2\}$ then the clock value of $x1$ will reach the value five before the clock value of $x2$ reaches the value two). The function $limit(x)$ refers to the value in which the clock x is compared with on the implicit transition tc_i (e.g., on the previous example, $limit(x1)$ =5 and $limit(x2)$ =2) (See the protocols *P1* and *P1-1* in Figure 4-6)

6. For each path pt of the new paths that are not of the form *PLP*.
   i. Start from the state s_0 or the state ds_i of each path and move towards the state source state of the implicit transition ss_i. (i.e., for all paths $PA(s_0,ss_i)$ *and* $PA(ds_i, ss_i)$)
   ii. For each transition t (ss, ds, tc, rc), **If** this transition t has a clock reset x $\in CK$ (e.g., the transition *a* and *d* in the protocol *P1* in Figure 4-6 are two examples of the transition t with clock reset *x1* and *x2* belongs to *CK*), **then** update the clock reset sequence of its destination state ($CRS(ds)$) by adding x as follow:
      a. **If** $CRS(ds)=\{\}$ (i.e., the clock reset sequence of the destination state is empty), then
         1. Add the clock x to the sequence $CRS(ds)$ (i.e., set x as the first element of the sequence $CRS(ds)$, $CRS(ds)=\{x\}$) . For instance, the update of $CRS(s_2)$ of the state s_2 in the protocols *P1-1* and *P1-2* in Figure 4-6).
         2. **If** $ds \neq ss_i$ (i.e., the state ds is not the end state of the path), then
            For each state s on the path from ds to ss_i,
            - Set $CRS(s)= CRS(ds)$. (e.g., the $CRS(s_3)=\{x1\}$ and $CRS(s_4)=\{x1\}$ in the protocol *P1-2* in Figure 4-6).

            Else

            - Assign this transition as an input to the state $(ss_i)_l$ where $Tc(t)_x$ checks the clock x

      b. Else
         1. **If** any transition of the path from the state ss to ss_i has clocks reset that belong to the sequence $CRS(ss)$, **then** delete these clocks from the sequence $CRS(ss)$.

2. Create f copies of the path from the state ss to ss_i, where f is the number of elements on the sequence CRS (ss). The new paths have the same start state (ss) but the end states are the copies of the state ss_i which are created in the step number one of the approach (e.g., for the transition d in the protocol $P1$-2 in Figure 4-6, f=1 because the number of the elements on CRS(s_2) is equal to one)
3. Update the CRS sets of the destination states of the new created transitions which have the source state ss and the constraints on their input transitions as follows:

❖ For b=1 to f+1, in which $t_b(ss, ds_b, tc_b, rc_b)$ is an output transition of the state ss (e.g., the transitions between the states s_2 and s_3 and s_2 and s_3 in the protocol $P1$-new in Figure 4-6).

- Set $CRS(ds_b)_b = x$, where $CRS(ds_b)_b$ represents the b^{th} element of the sequence $CRS(ds_b)$ (e.g., in the protocol $P1$-new in Figure 4-6, the value of b is equal *one* in the state s_3 and as a result $x2$ is inserted as the first element on the CRS(s_3) but b equal to the value *two* in the state s_3 and as a result it is inserted as a second element after $x1$ on CRS(s_3)) .
- Set $tc_b = [tc \wedge ((CRS(ds_b)_{b-1} > (limit\ (CRS(ds_b)_{b-1}) - limit(CRS(ds_b)_b)) \wedge ((CRS(ds_b)_{b+1} < (limit(CRS(ds_b)_{b+1} - limit(CRS(ds_b)_b)]$. This step updates the constraint on the transition based on the order of clocks on the CRS. The first part tc represent the constraint before updating and the second part constructs the constraint by setting the clock number b-1 in the order in CRS of the destination state greater than the difference between it and the clock number b and the third part set the clock number b+1 to be less than the difference between the limits of the clock b+1 and the clock number b. For example in the protocol $P1$-new in Figure 4-6 the state s_3 has the $CRS(s_3)$={$x2$,$x1$} and the resulted time constraint is $x1$<3 which is resulted by the third part of the previous updating formula.
- **If** ($ds_b = ss_i$) (i.e., the end of the path) **then**
  - Create f-1 copy of the transition t_b.
  - Assign each $(t_b)_l$ transition as an input to a state $(ss_i)_l$ where $Tc(t_b)_l$ checks the clock x_b.

- Update the time constraint on each transition $(t_b)_l$, set $tc_b = tc_b \wedge x_b < limit(x_b) \wedge (CRS(ds_b)_1 > limit(CRS(ds_b)_1) \wedge CRS(ds_b)_{b-1} > limit(CRS(ds_b)_{b-1}$. This update is based on the clock x_b that is checked by the $Tc(t)_l$ on the output implicit transition of the state $(ss_i)_l$ which is the destination state of this transition $(t_b)_l$.

- **If** $rc_b \notin CK$, create an additional copy of the transition t_b and assign it as an input for the state (sa) which is created in step 3 with time constraint $tc_b = tc_b \wedge x_b > limit(x_b) \wedge (CRS(ds_b)_1 > limit(CRS(ds_b)_1) \wedge CRS(ds_b)_f > limit(CRS(ds_b)_f$.

- **Else if** $(ds_b \neq ss_i)$, **then**
  - for each state s on the path from ds_b to ss_i,
    - α Set $CRS(s) = CRS(ds_b)$
    - α Set the path starting from the state ds_b to the state ss_i as pt and repeat from step 6.ii (see the protocols *P1-2* and *P1-new* in Figure 4-6).

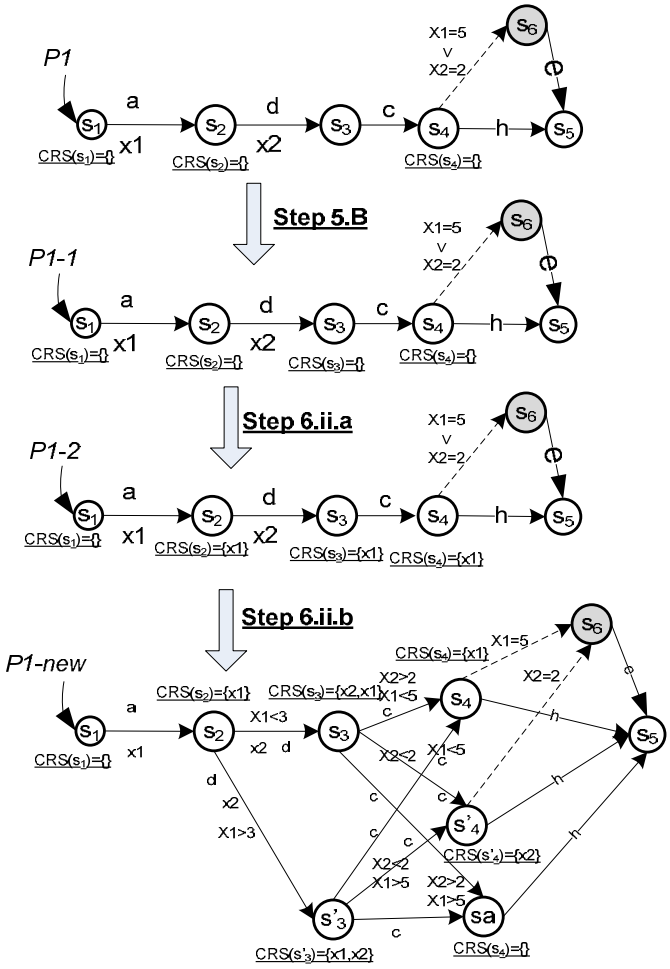

Figure 4-6: Protocol *P1* and its equivalent protocol *P1-new* after applying the steps 5 and 6.

7. For each path *pt* of the new paths that are in the form PLP, where the path enters the loop through state (s_{il}) and exit through state (s_{ol}), we have two cases.
   i. **Case one**: **if** all the transitions on the loop do not contain any clock reset that belongs to *CK*. Then apply **step 6** after considering the loop as one transition, its source state is s_{il} and destination state is s_{ol} (See Figure 4-7).

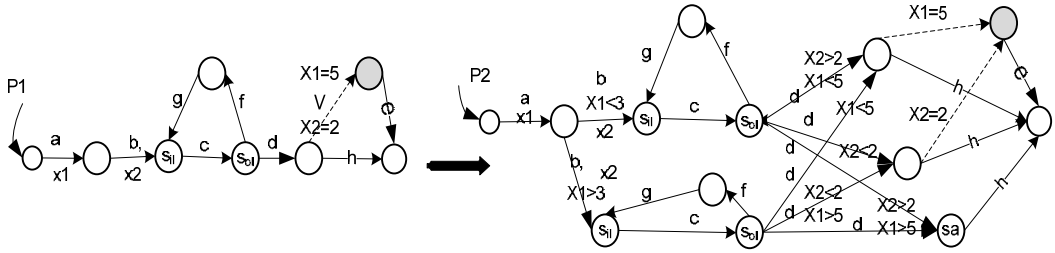

Figure 4-7: Protocol *P1* without clock reset in the loop and its equivalent protocol *P2*.

ii. **Case Two**: the loop has transitions with a clock reset that belongs to *CK*

a. **If** these transitions are only in the path starting from the state s_{il} to the state s_{ol} on the loop (i.e., the clock reset **is not** on the path from s_{ol} to s_{il}) then

- Apply the step 6 and add to the **step 6.ii.b.2** a copy of the path starting from s_{ol} to s_{il} for the new *f* copies of the path (See Figure 4-8).

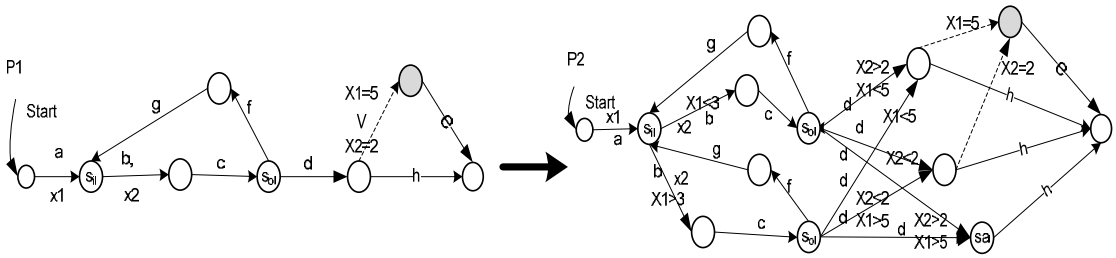

Figure 4-8: Protocol *P1* with a clock reset (*x2*) in the loop in the path between the states s_{il} and s_{ol} and its equivalent protocol *P2*.

b. **Else if** these transitions are in the path starting from the state s_{ol} to the state s_{il} on the loop **then**

- Create a path from the state s_{ol} to the new state $s_{\theta l}$ equivalent to the loop starting form s_{ol} to the same state s_{ol} and delete the transitions on the path $PA(s_{ol}, s_{il})$.

- Create a path from the state s_{ol} to the state ss_i equivalent to the path $PA(s_{ol},ss_i)$.
- Let t (s_{ol} , s, tc, rc) be the first transition on the path $PA(s_{ol},s_{il})$.
- Create a copy of the transition t where the source state of the new transition is s_{ol} and the destination state is s (See Figure 4-9).
- After these operations we got a new path with a loop and another path without loop.
  - **If** clocks reset rc in t does not belong to CK, then use **case two** for the path with the loop and **step 6** for the other path.
  - **Else if** clocks reset rc in t is belong to CK, then for the new path that has a loop , use **case two** . After the execution of this step, we get f loops where f is the number of clocks reset before the state (s_{il}) of the loop.
    - ✓ From each state (s_{ol}) of each loop of the *f loops*, create f copies of the transition t.
    - ✓ The destination states for these copies are the input states (s_{il}) of the other *f loops*.
    - ✓ Update the time constraint for the new created transition t to be the same as the time constraint of the input transition to the state (s_{il}) (See Figure 4-10).
  - For the transitions of the path outside the loop, apply **step 6.ii**.

8. Delete t_i the source state ss_i of the implicit transition.

End

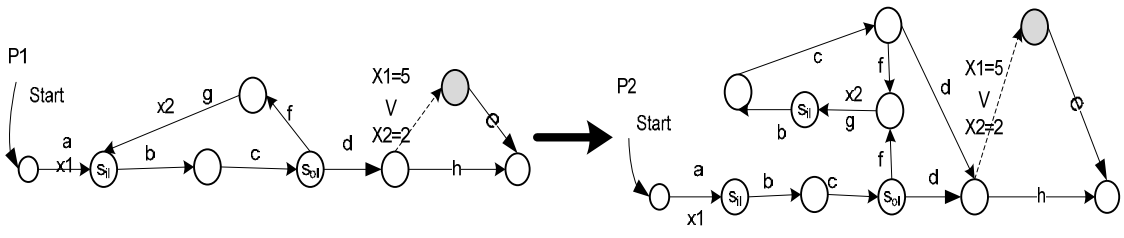

Figure 4-9: Protocol *P1* with clock reset in the loop and its equivalent protocol *P2* (this clock reset is not on one of the output transitions of the state s_{ol} and in the path between s_{ol} and s_{il}) .

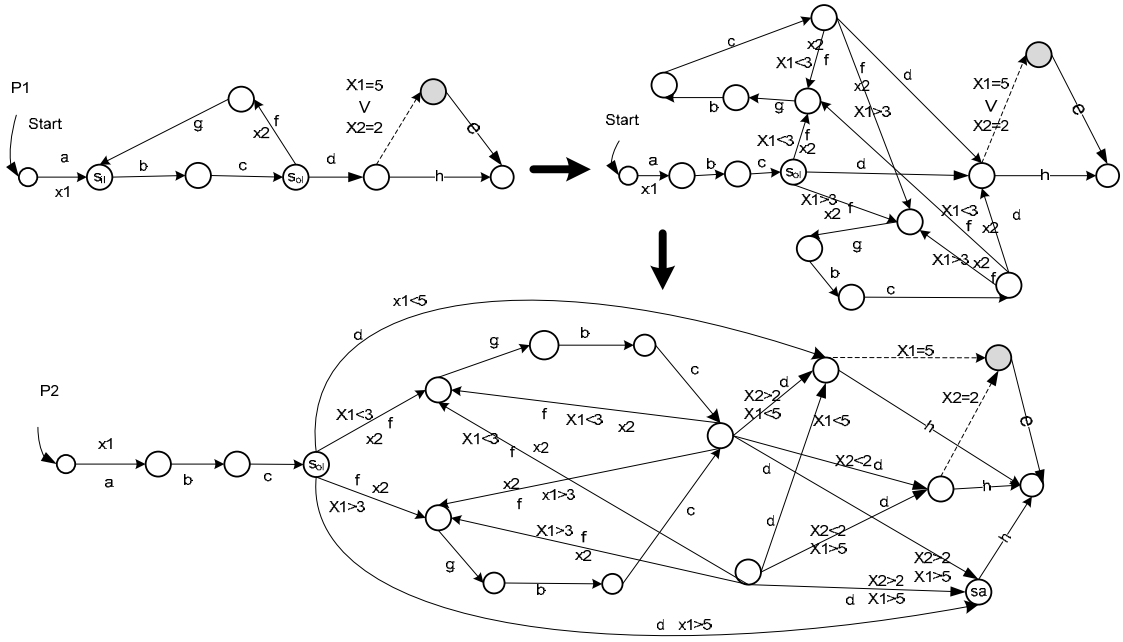

Figure 4-10: Protocol *P1* with clock reset in the loop and its equivalent protocol *P2* (this clock is on one of the output transition of the state s_{ol} and in the path between s_{ol} and s_{il}).

4.1.2 Analysis and proof

The correctness of this approach can be verified by proving that the output protocol represents the same semantics of the input protocol. The approach can be divided into two main operations, (1) The overlapping separation operation, (2) Paths duplication operations. As shown in the approach, the overlapping separations process does not change the time constraints and preserves the same set of the timed conversation of the original protocol. As a result, this step does not change the semantics of the protocol. The second operations are based on the following simple rule: any explicit transition $t(s,s',m,tc,cr)$ can be separated to two transitions

$t1(s,s',m, x=>v_1 \wedge tc,cr)$ and $t2(s,s',m, x<v_1 \wedge tc,cr)$ where $x \in CR$ and $v_1 \in \mathbb{R}$. The

timed conversation of the original protocol can be presented by the new one after applying these operations.

The complexity of this algorithm depends on the number of clocks that are checked on the implicit transition and the number of transitions from the start state to the source state of the implicit transition. Therefore, the complexity is O($n \times m$) where n is the number of checked clocks on the implicit transition and m is the number of the transition for the longest path from the start state to the input state of the implicit transition.

4.2 The conversion approach

This approach is used to convert a business protocol with implicit transition to a new one without implicit transitions [65].

Begin (**Input**: MCTBP, **Output**: MCETBP))

1. Check the destination state of the implicit transition, and if it has input transitions other than the implicit transition, then create a copy of this state with its input and output transitions without copying the implicit transition. This operation is used because of the fact that deleting implicit transition and merging its source state and destination state, while preserving the protocol's semantics, cannot be done if the destination state has other inputs. This step preserves the semantics of the protocol because all the execution paths before this step can be executed after it (See Figure 4-11).

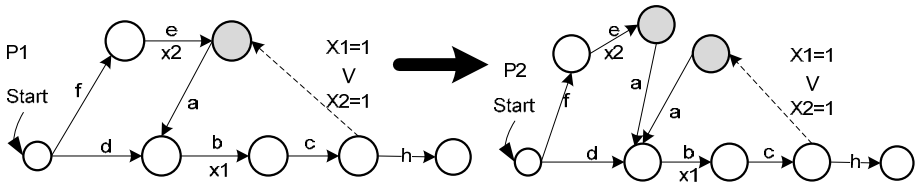

Figure 4-11: Business protocol *P1* and its equivalent protocol *P2* after the execution of step 1.

2. **If** the time constraints on the implicit transition is not in a disjunctive normal form (i.e., $Tc(ti) = \bigcup_{i=1}^{n} Tc(ti)_i$ where n is the number of disjunction terms), **then**

rewrite it to be in this form. In Figure 4-11, $Tc(ti)=\{x1=1 \vee x2=1\}$ is already in disjunctive normal form. By using this operation, one knows which clock values trigger the implicit transition and the time constraints assigned to explicit transition after removing the implicit transition.

3. Check the presence of clocks reset on the implicit transition, we will call this clock "implicit clock". After checking, we have three possible cases.

Case one: There is no clock reset (i.e., there is no clock reset on the implicit transition), such as in the previous example which is shown in Figure 4-11.

A. Apply the previous approach "***separation approach***" (Section 4.1.1) for replacing the implicit transition (ti) with n implicit transition $(ti)_l$ where $l=1$ to n (where n is the number of the disjunctive terms on the implicit transition constraint).
B. Create n copies of the destination state of the implicit transition and its output transitions.
C. Update the time constraints on the output transitions of the destination state of the implicit transition $(ti)_l$ by time interval constraints restrict its triggering to clock values greater than the triggering instance values on the implicit transition(e.g., if $(ti)_l=\{x1=1\}$,then the time constraints is $x1\in[1,\infty[$).
D. Update the time constraints on the output transitions of the source state of the implicit transition $(ti)_l$ by time interval constraints restrict its triggering to clock values less than the triggering instance values on the implicit transition(e.g., if $(ti)_l=\{x1=1\}$,then the time constraints is $x1\in[0,1[$).
E. Delete the implicit transition.

Case two: There is a clock reset on the implicit transition and this clock is **not** included in the time constraint of the implicit transition *Tc*(*ti*).

A. Use the previous approach "***implicit transition constraint separation***" for replacing the implicit transition (*ti*) with *n* implicit transition $(ti)_l$ *where l=1 to n*.

B. Create *n* copies of the path from destination state of the implicit transition to the destination state of the transition *t*(*ss*,*sd*,*tc*,*rc*) that has a time constraint (*tc*) contains the implicit clock (e.g., *tc*={*x3*=1} and *x3* is the clock reset on the implicit transition $(ti)_l$).

C. For each implicit transition $(ti)_l$, **if** there is an explicit transition *t*(*ss*,*sd*,*tc*,*rc*) , where *rc* contains clocks reset and these clocks are in the time constraint of the implicit transition $(ti)_l$) in one of the created paths in the previous step.

   I. Create a copy of all the loops from the source state of this implicit transition $(ti)_l$ to the same state, and a copy of the output transitions of the state which prior this source state.

   II. Rename the explicit clock (e.g., *x1* becomes *x1*') on the original implicit transition constraint and on *tc* on the new path.

   III. Update the time constraint *tc* by replacing the implicit clock (e.g., *x3*) by its equivalent from the implicit transition time constraint, Figure 4-12 shows an example after the execution of the **steps I, II, and III**.

D. **Else**, update the time constraint *tc* by replacing the implicit clock (e.g., *x3*) by its expression deduced from the implicit transition time constraint $tc(ti)_l$ (e.g., if $tc(ti)_l$ of implicit transition is {*x1*=2}, and the tc is x3=1, then tc becomes x1-2=1).

E. Execute the steps C, D, and E of the case one**.**

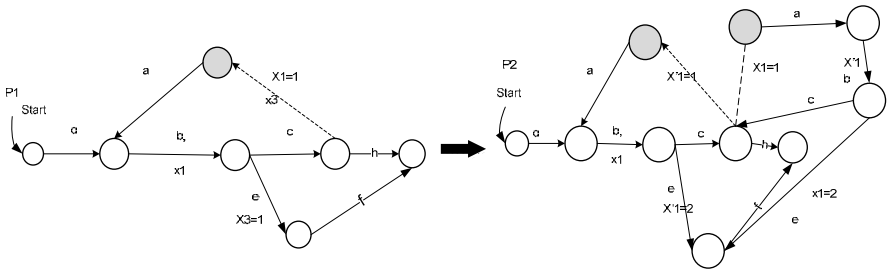

Figure 4-12. Protocol *P1* and its equivalent *P2* after the execution of steps I, II, and III.

Case three: There is a clock reset on the implicit transition and this clock is included in time constraint $Tc(ti)$.

A. Apply the "***implicit transition constraint separation***" approach without considering the implicit clock (i.e., remove the implicit clock from the implicit transition constraint before applying the separation approach).
B. For each implicit transition ti_l (ss_l,ds_l,tc_l,rc_l), do the following steps.
C. Starting by the state ds_1, create paths equivalent to any loop from this state to itself with new end states ds'_1 and new implicit transition $(ti')_l$ with $tc(ti')_l = tc(ti)$ and remove the loop by removing the output transitions of the state ds_l which were in the loop (See Figure 4-13)

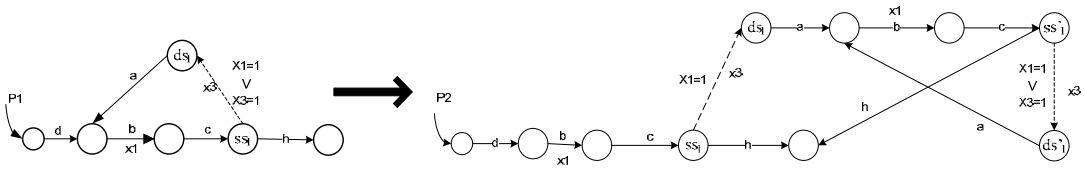

Figure 4-13. Protocol *P1* and its equivalent *P2* after the execution of steps C.

D. Check the paths from destination state of the implicit transition $(ti)_l$ to source state of the new implicit transition $(ti')_l$
   I. **If** it has an explicit transition t'(ss', sd', tc', rc') where rc' has clock reset (e.g., *x1*) and these clocks are in the time constraint $(tc_i)_l$ of the implicit transition $(ti)_l$, **Then** rename the explicit clock (e.g., *x1*) on the transitions t' and $(ti')_l$ and update the time constraint $tc(ti')_l$ by replacing the implicit clock (e.g., *x3*) by its equivalent from the implicit transition time constraint(see Figure 4-14).

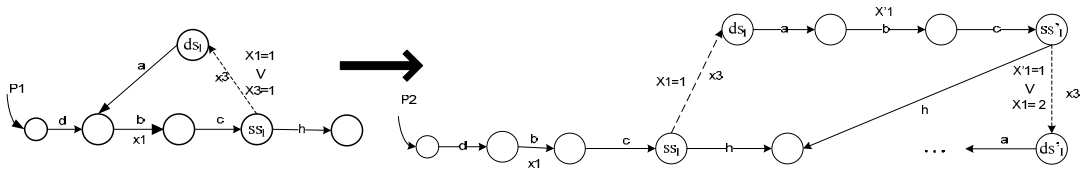

Figure 4-14. Protocol *P1* and its new form (*P2*) after the execution of steps D.I.

   II. **Else**, update the time constraint $tc(ti')_l$ by replacing the implicit clock (e.g., *x3*) by its expression deduced from the implicit transition time constraint

$tc(ti)_l$ (e.g., if $tc(ti)_l$ of implicit transition $(ti)_l$ is $\{x1=2 \vee x3=1\}$, then $tc(ti)_l$ becomes $\{x1=2 \vee x1-2=1\}$.

E. For the implicit transition $(ti)_l$, apply the implicit constraint separation approach.

F. After the separation process, for each new implicit transition $(ti)_{lz}$ of the implicit transition $(ti)_l$, compare the time constraint $tc(ti)_{lz}$ on the implicit transition with $tc(ti)$.

I. **If** $Tc\ (ti)_{lz} = Tc\ (ti)$, then merge the two implicit transitions and update the input transitions of the $(ti)_{lz}$ by resetting the value of k to start from zero.

II. **If** the values of the constraints limit are different but they check the same clocks (i.e., $x(tc(ti)_{lz})_x = (tc(ti))_x$, **then**

The value of these constraints can be presented based on the two values of the clock. The constraint will be $x1 \in [a+k*(b-a), \infty[$ with $x1=a$ in the first constraint $tc(ti)$, $x1=b$ in the next constraint $tc(ti)_{lz}$, *and* $k=\{0,1,2,\ldots\}$.

i. Merge the two implicit transitions,

ii. Update all the time constraints in the path after merging the two implicit transitions by replacing the limit value of the clock $(Tc\ (ti))_x$ by k which presents the **arithmetic sequence.**

III. In case $(tc(ti)_{lz})_x \neq x(tc(ti))_x$ then repeat the same procedure from step B in this case with the new implicit transition $(ti)_{lz}$.

G. Execute steps C, D, and E of **case one** (See Figure 4-15)**.**

END

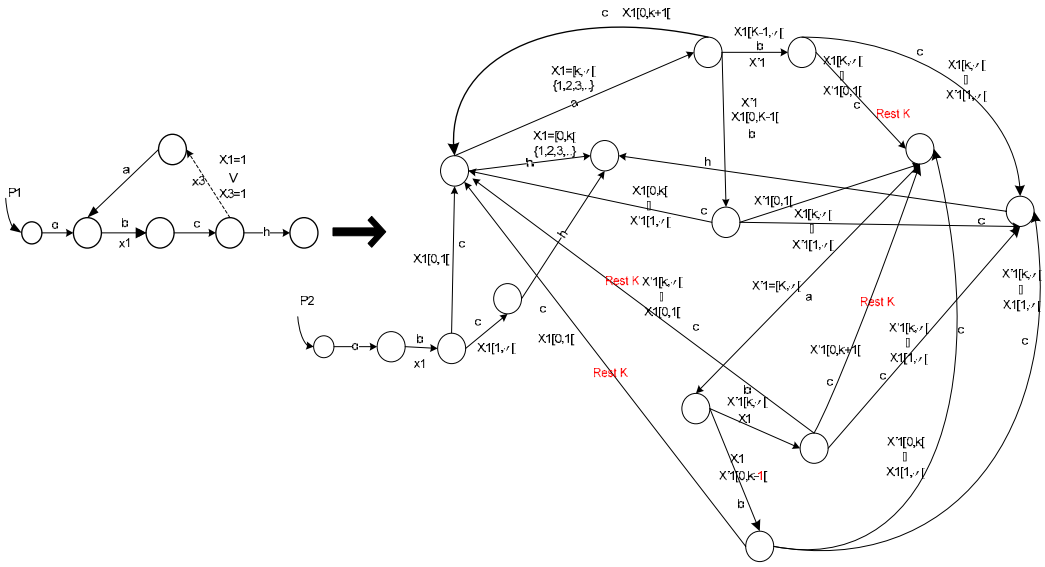

Figure 4-15: Protocol *P1* with clock reset (*x3*) and implicit time constraint (*x1*=1 ∨ *x3*=1) checks this clock and its equivalent protocol *P2* after the execution of the algorithm.

4.2.1 Analysis and proof

Deleting implicit transition and merging its source state and destination state, while preserving the protocol's semantics, cannot be done if the destination state has inputs rather than the implicit transition. This operation is performed by step 1 which preserves the semantics of the protocol by preserving all the paths of the protocol. Step two is for rewrite the time constraints on the implicit transition. By using this operation, one knows precisely which clock values trigger the implicit transition and the time constraints assigned to explicit transition after removing the implicit transition. Step 3 manipulates three different cases of the implicit transitions. The first case deals with implicit transition **without** clock reset. In this case, there is no transition on the protocol has a time constraint depends on the time of the triggering of the implicit transition. The next transitions after the implicit tran-

sition are the only affected transition after the triggering of the implicit transition. Therefore, in this step, the separation technique is applied and then a new copy of the next following transition for each one of the implicit transition is create and a time constraint based on the checked clock on the implicit transition is assigned.

The second case deals with implicit transition **with** clock reset (implicit clock) on it and this clock is **not** checked in the time constraint of the implicit transition constraint *Tc*(*ti*). This implicit clock will be checked in one or more on the time constraints of the following transitions but it will not be checked on this implicit transition. Consequently, the approach duplicates not only the next transition of the implicit transition but also all the following transitions until reaching the transition which has a time constraint checks the implicit clock. Then, it updates the time constraint on the new paths based on the constraint on each implicit transition.

The third case deals with a clock reset on the implicit transition and this clock is included in the time constraint of the implicit transition *Tc*(*ti*). This case is the most complicated case because the approach must preserve the semantics of the implicit clock reset and it□s checking on the implicit constraint after deleting the implicit transition. To replace the implicit clock with explicit one, we must know which explicit clock triggers the implicit transition on its last triggering. Therefore, the approach will present all the possible paths of the loop starting from the implicit transition and ends with the same implicit transition in the next iteration with the new time constraint. Each path of the new created paths will be treated separately based on the time constraints on its two implicit transitions. The time constraint on the explicit transitions on each path after deleting the implicit transitions is based on the time constraints on the two implicit transitions in the path. For updating the time constraint of the explicit transition which are affected by the deleting of the implicit transition, the value of the limit of each clock on the two implicit transitions is compared to see if this value is rest constant for each iteration or different form one iteration to the other. If this value is always the same, the time constraint on the explicit transition will base on this constant value. But if the value is different, the set of values will be presented as an **arithmetic sequence** and the constraints will be based on this sequence. Therefore, the implicit clock is replaced by the previous clock which triggers the implicit transition for each path and this is based on the fact that the first triggering of the implicit transition cannot be done by the implicit clock because the first reset of the implicit clock will be in the first triggering.

The complexity of this algorithm for the first case is O($m \times n+L$) where n is the number of checked clocks on the implicit transition, m is the number of the transition for the longest path from the start state to the input state of the implicit transition, and L is the number of output transitions from the destination state of the implicit transition. The complexity for the second case is O ($m \times n+L \times H$) where H is the number of transitions for the longest path from the destination state of the implicit transition to the explicit transition that is triggered based on a time constraint contains the implicit clock. The complexity of the third case is O($f \times$ ($m \times n+L \times H$) $\times n$) where f is the number of transitions on the loop that is reset using clocks from the implicit transition constrains.

Chapter 5 .WEB SERVICES ACCESS CONTROL

Modeling and analyzing Web services after taking into account the security issues has become an urgent necessity. Accordingly, this chapter presents a set of Web services security concepts and shows the approaches in which these security requirements are modeled and analyzed. Firstly, it explores the Web service access control models. Secondly, an informal scenario and the proposed architecture are explained. Thirdly, the role of the ontology in the analysis is clarified. Finally, the related work are listed and discussed.

5.1. Web Services AC models

Due to the open nature of the Web, its security becomes a crucial requirement. The main goal of access control is to restrict the access of the Web resources to specific users with specific requirements (credentials). Because Web services are part of the Web with special properties, development of suitable access control models able to restrict access to Web services to authorized users is an important issue (i.e., access control policies can be seen as conditions in which a Web service provider decides to restrict the set of users who may access the functionalities offered by her Web service**)**. Traditional Web security approaches and technologies commonly adopted for Web sites and traditional access control models are not satisfactory for securing Web services (Bertino et al. [29]). Therefore, there is a need for new approaches for enforcing Web services access control policies.

The eXtensible access control markup language (XACML) [101] is a reference access control model proposed by OASIS. It presents mechanisms to define access control policy. Different control policies and be defined using these mechanisms under different security requirements. Figure 5-1 shows the XACML architecture in which the component □recourses□ corresponds to Web services. This architecture has three main components, PDP (Policy Decision Point), PEP (Policy Enforcement Point) and PAP (Policy Administration Point). The PAP writes policies and policy sets to the Policy repository. Context handler do format exchange among XACML components. With the control of XACML, when a requester re-

quests to access a resource, PEP takes control to determine whether the request should be allowed or rejected. During this process, the Policy Information Point (PIP) collects attributes such as resource name, the requester's role, system time, and operation type. The collected attributes are then sent to PDP, which makes decision by checking whether the attributes fulfill the policies associated with the invoked web service. The final decision is returned to PEP. If the request is allowed, the requester can access the service. Otherwise, the request is rejected.

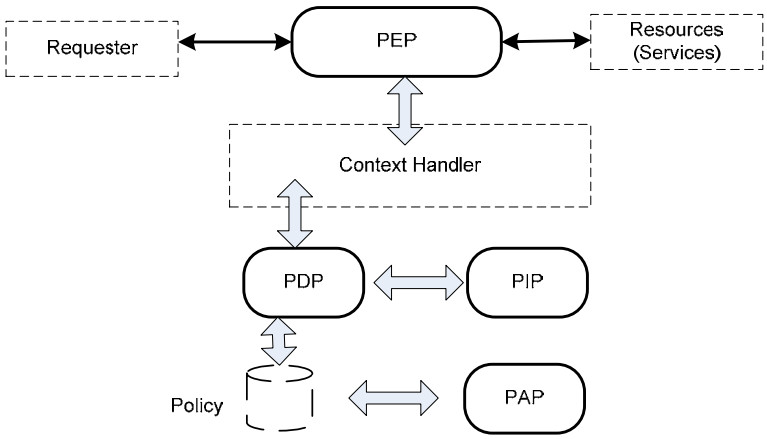

Figure 5-1: A XACML architecture.

Most of the existing Web services access control approaches, including the approach proposed by Ardagna et al. [7], assume a single operation model for Web services where the invocation of operations are independent from each other. But, Most of Web services are composed of a set of dependent operations. These operations are accessed in a particular order specified by the business protocols of the services (Benatallah et al.[19], Berardi et al.[22], Aalst and Hofstede [136]). Therefore, Mecella et al. in [100] present a Web service architecture for enforcing access control policies for web services that are composed of a set of dependent operations (See Figure 5-2).

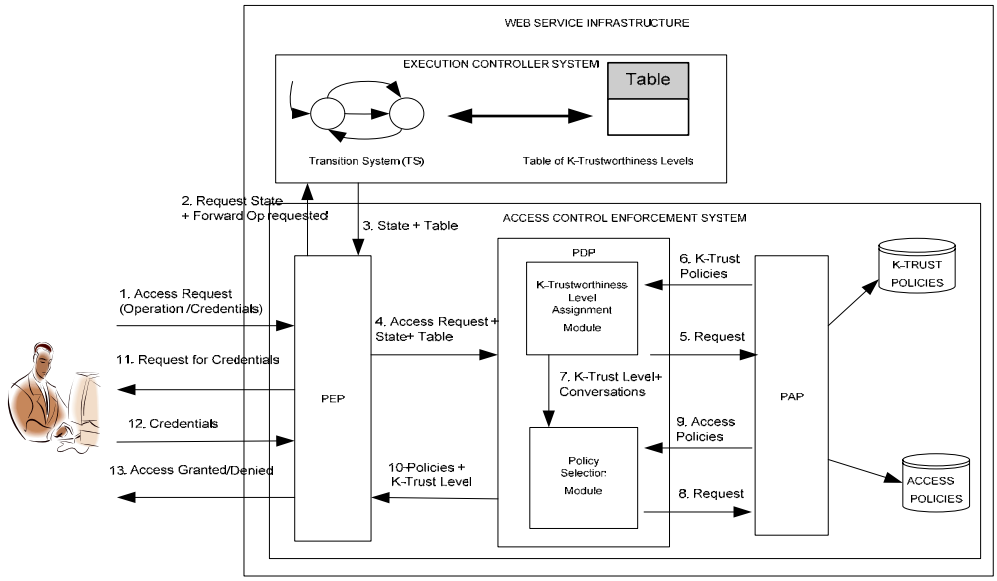

Figure 5-2: Web service architecture for enforcing access control policies proposed by Mecella et al. in [100].

The system architecture consists of two basic components, the access control enforcement system and the execution controller system (ECS). The access control enforcement system contains the PEP, PDP, PAP, K-trust policies, and access policies which makes it compliant with the XACML standard. The PEP receives the client requests with the operations names and credentials and passes it to the ECS which receives the current states of the conversation and the table. Then, it reformulates the access request by adding information about the current state of the conversation and the table and sends it to the PDP. A trustworthiness level represents the length of a conversation, from a given state s in the transition system that leads to a final state. The PDP consists of two modules, the k-Trustworthiness Level Assignment (TLA) module and the Policy Selection (PS) module. The TLA uses the table to select the trustworthiness levels $k_1, \ldots, k_n$ contacts with the PAP to get the k-trust policies associated with trustworthiness levels $k_1, \ldots, k_n$. Then, it evaluates if the credentials provided by the client in the request satisfy one of the policies. If this is the case, the client is assigned to the trustworthiness level k_i associated with the k_i-trust policy he/she is compliant with. The PS receives the associated conversations after assigning trustworthiness level k_i and asks the PAP for the access control policies related to the operations constituting the conversations that

the client may engage with the service on the basis of the assigned trustworthiness level k_i. The selected policies are combined to obtain the corresponding conversation access control policy. After that, the PEP receives the policies with k_i. Then, the PEP sends a request to the client asking for the credentials required by the policies and evaluates them against the policies. If the check is positive, the client can perform any operation in the conversations related to the trustworthiness level k_i. When the client submits a request to perform an operation, which does not belong to the allowed conversations, the PEP contacts the PDP, which assigns a new trustworthiness level to the client. In this case, the PEP does not send again the table of trustworthiness levels, but only the state of the conversation with the client, which is necessary to select from the table the trustworthiness levels associated with that state because it stores a copy of the table of trustworthiness levels and the level k_i assigned to the client

Because our work is based on Web service conversation model, there is a relation between our work and the work which is done by Mecella et al. in [100]. Both of us present the Web service access control as a state transition system but they do not consider the business protocol of the consumer. They only consider consumer credentials. There are some situations where the credentials of the consumer satisfy the AC but based on the business protocols there is no conversation can be completed. Therefore, there is a need for a preliminary step for checking the compatibility after including the AC. This step can be integrated with the standard AC model and the model which is presented by Mecella et al. in [100].

To the best of our knowledge, in most of access control enforcement models, checking of the compatibility and replaceability after assigning AC is not included. We will merge all the previous research issues by including the compatibility and replaceability checking box in the enforcement model and talking time into account. This step can help basically in two scenarios:

- The consumer will invoke the service which is compatible with him in terms of message exchange, time, and access control, (i.e. based on the set of credentials with the consumer he can access the service which accept his credentials).
- With respect to replaceability, service can replace another service with the guarantee that it has the same access control and it is compatible with all the services which are compatible with the replaced service.

5.2. Informal scenario and architecture

Checking compatibility and replaceability between two Web services after assigning the access control policy is important in many scenarios. For instance, the scenario which can be happened with a consumer searching for a Web service:

1- Consumer searches the registry of services (e.g. UDDI) and found the service with required operations and gets its binding information (See step 1 and 2 in Figure 5-3).
2- The service provider asks for the credential ***C1*** and the consumer sends it (See steps 4 and 5 in Figure 5-3).
3- A set of request/response interaction is performed between the service and the consumer without problems.
4- In the step ***i,*** the service asks for the credential ***Cn*** to access its operation and the consumer has not this credential (See steps *i* and *i*+1 in Figure 5-3).
5- As a result, the interaction is terminated and the consumer starts again from the beginning using another service (step *i*+2).

This scenario continues until the consumer gets the service compatible with him in terms of access control and accepts his credentials which is a problem because of the wasted time and the credentials which are provided by the consumer. To solve the problem in this scenario, at the beginning, service provider shows the access controls for the service operations by two approaches. In the first approach, a description of the policies for each operation is presented independently on each others. But this approach is not useful in the situation where service consists of more than one operation and these operations are dependent on each others. Therefore, the second approach proposes to present the AC as state transition model indicating the relation between these set of AC, as the model presented by Mecella in [100]. But this approach is enforced without considering the business protocol of the consumer. They only consider consumer credentials. There are some situations where the credentials of the consumer satisfy the AC but based on the business protocols there is no conversation can be completed. Therefore, there is a need for preliminary step for checking the compatibility and considering the AC. Figure 5-4 shows an informal scenario with a modified architecture to overecome the problem of the traditional AC models. This step can be integrated with the standard AC model and the model which is presented by Mecella in [100]. Also replaceability analysis is important in a situation which

needs satisfaction between the two protocols in terms of access control model. For instance:

- Service provider wants to replace his old service with a new one which supports the same conversations with the same AC (i.e. the new service has more functionality than the old one but all the functionalities of the old one must be 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 AC (i.e. the new service has more quality of service).

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. Therefore, we merged the time business protocol with AC model in checking the compatibility and replaceability analysis.

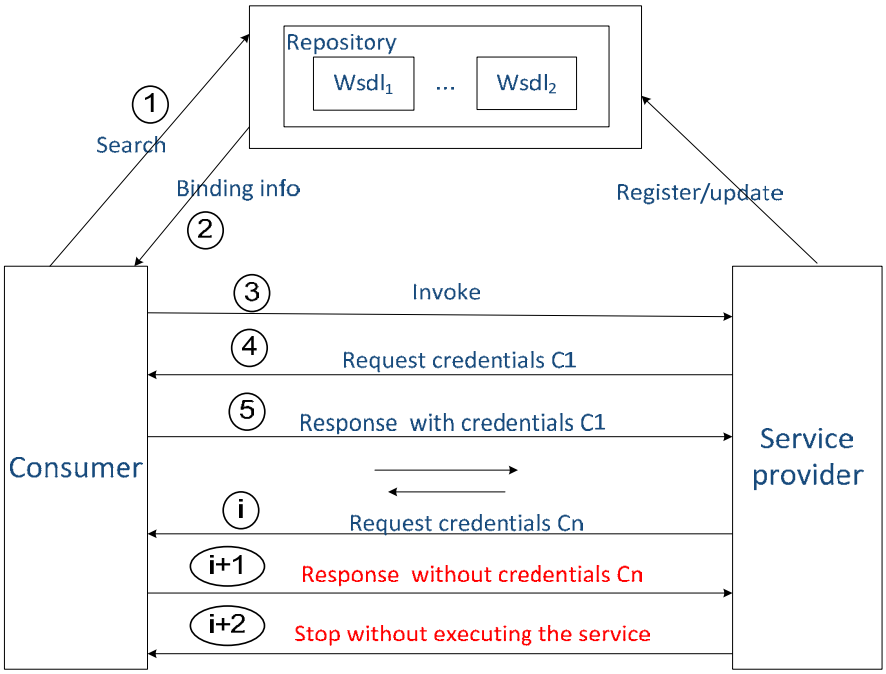

Figure 5-3: Informal scenario shows the problem of the traditional AC models.

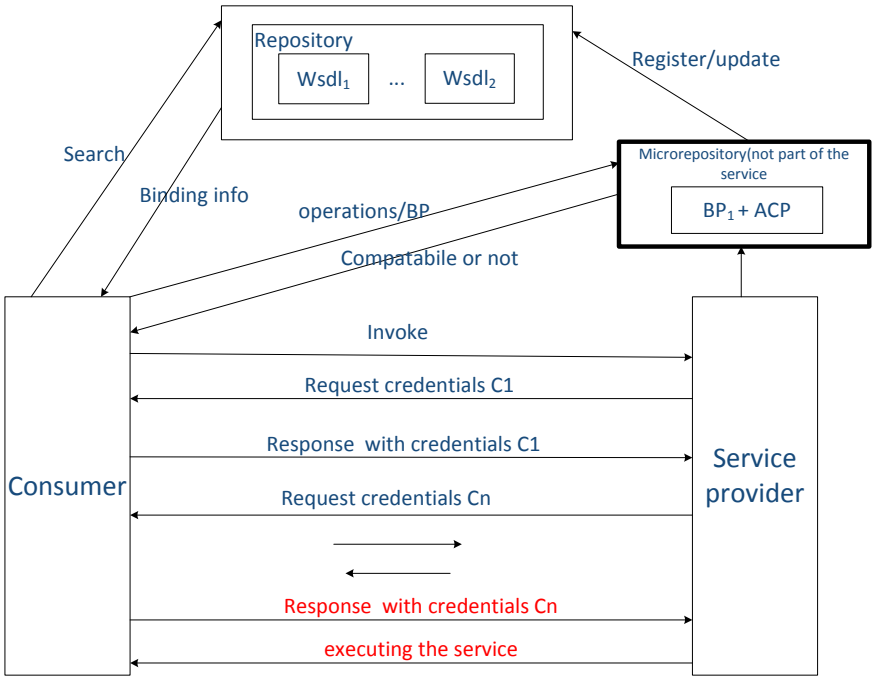

Figure 5-4: Informal scenario with a modified architecture to overecome the problem of traditional Web services AC models.

The proposed architecture is shown in Figure 5-5. This access control enforcement system (ACES) model is composed of PEP, PDP and PAP which make it compliant with the XACML standard. The PEP realizes the interface with clients and the checker unit. The checker has a set of tools for checking the compatibility and replaceability analysis using business protocols augmented with the access control policies before the invocation of the service. It can also be used to keep track of the state of the conversation between the consumer and the service during the interaction (i.e. after invoking of the service). Furthermore, it uses the ACP ontology in calculating the subsumption during the checking process. The PEP intercepts all the access requests submitted by clients to access an operation that uses policy-based access management, the PEP will describe the user's attributes to other entities on the system. The Policy PDP has the job of deciding whether or not to authorize the user based on the description of the user's attributes provided by the PEP. PAP can provide many enterprise SOA policy administration capabilities but the end result of the policy administration point is to store or distribute policy updates. As shown from the architecture, our contribution is mainly the checking unit

which can be used with many access control enforcement system models. The integration of these modifications with any access control enforcement system does not require a big modification of these existing models and can be integrated easily to them.

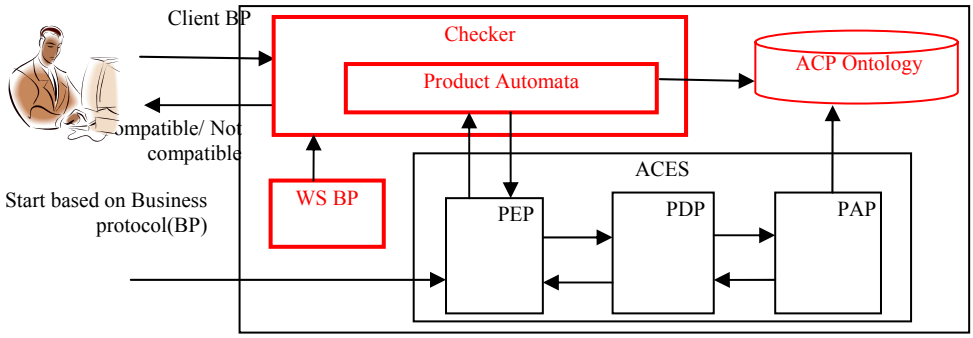

Figure 5-5: Access control architecture model with compatibility and replaceability checking tools.

5.3. Ontology

In our model, the ACP and credentials are presented in forms of ontology concepts. There are two main advantages of using ontology in policy specification and management [103]:

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) [9] are a set of knowledge representation languages that can be used to represent the knowledge of an application domain in a structured and formally well-understood way. The important notions of the domain are described by descriptions (i.e., they are described by atomic concepts and atomic roles) and equipped with a formal, logic-based semantics. In the syntax of description logics, concept expressions are variable-free and each concept expression denotes the set of all individuals satisfying the properties specified in the expression. Concepts properties are expressed by roles**.** Semantically, a concept is interpreted as a set of individuals and roles are interpreted as sets of pairs of individuals. The

description can use set of Boolean constructors such as conjunction (⊓) and negation (¬), as well as the existential restriction constructor (∃AR.CN), where CN is an atomic concepts and AR is an atomic role (e.g., ∃hasVisa.Human), value restriction constructor (∀AR.CN) (e.g., ∀hasChild.Person), and the number restriction constructor (≥ n AR) (e.g., ≥ 5 haschildChild).

The description logics provide a well-defined semantics and powerful reasoning tools such as tableau-based algorithms. Therefore, it is candidates for ontology languages. Description logics had a strong influence on the design of OWL (Ontology Web Language), particularly on the formalization of the semantics, the choice of language constructors, and the integration of data types and data values. OWL is the World Wide Web Consortium (W3C) recommended ontology language for the semantic Web, and exploits many of the strengths of description logics, including well defined semantics and practical reasoning techniques. Description logics can be used 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. The access control policies are presented as concepts describe set of credentials as indi-

viduals. We will perform subsumption (⊆), union(∪), and intersection(∩) operations on the ontology during our algorithms.

Presenting the AC of Web services as ontology will enable 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.

5.4. Related work

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 (Ferraiolo et al. [67]), the RBAC96 model (Sandhu et al. [123]), WS-AC1 (Bertino et al. [30]), the RT model (Li et al. [95]) and conversation-based Web services access control model (Mecella et al. [100]). In a parallel, researchers have developed policy languages for access control. These include XACML (Moses [101]), Security Assertion Markup Language (SAML) [80], Ponder (Damianou et al. [53]), WS-Policy (Bajaj et al. [10]) and finally to semantic Web based languages such as Rei (Kagal et al. [85]), DARPA agent markup language for services (DAML-S), and KAoS (Tonti et al. [131]).

The XACML and SAML are two major standards. The XACML is an XML framework for specifying access control policies for Web-based resources in general and it has been extended to specify access control policies for Web services. The SAML defines an XML framework for exchanging authentication and authorization information for securing Web services. Other standards are the WS-Policy and the WS-Security. WS-Policy is used to describe the security policies in terms of their characteristics and supported features (such as required security tokens, encryption algorithms, privacy rules, etc.). WS-Security is a specification for secur-

ing SOAP messages using XML encryption and XML signature standards and attaching security credentials thereto.

Wonohoesodo and Tari [142] propose two access control models, SWS-RBAC (for single services) and CWS-RBAC (for global services or composite services). They divided the roles to global role and local role and the client is assigned to access composite service with only global role, which is mapped onto local roles of the service providers of the component Web services.

Bertino et al. [30] presents a flexible access control model (WS-AC1). The AC polices are specified as a conditions on the values of the identity attributes and service parameters that a consumer must provide to invoke the service. This model has negotiation capabilities. These capabilities are related to the identity attributes and service parameters. During the negotiation process, the consumer is guided toward an access control request compliant with the service description and policies.

There are many works in the access control enforcement in business process workflow (Hwang et al. [83], Paci et al. [106]). Hwang et al. [83] present a Web service selection approach in workflow, composed of set of tasks that dynamically chooses a performer for each task in the workflow satisfying all access constraints currently and increase the chance of completing the entire process in the future. They disallowed delegation sequences to model many types of access control constraints. Their access control enforcement process supposes that there are some steps in the workflow are performed manually. Paci et al. [106] concentrate on the specification of the authorization information associating users with human activities (activities in WS-BPEL business process which requires human interaction) in WS-BPEL business process and authorization constraints by using a role-based access control model for WS-BPEL.

Skogsrud et al. [126] presented a model for trust negotiation policies for Web services. They assign for each role (for instance, customer, reviewer, etc. in bookshop service) the set of operations and the required credentials. State transition graph is used where states present the roles and transitions present the operations and the policy. This model is used in the access control trust negotiation during the interaction between the services and the consumer. Another example for trust access control model for Web service is the model presented by Coetzee and Eloff [50]. This

model grants and adapts permissions assigned to both Web services requesters and their respective users based on the current context of trust relationship that exist.

Access control models for Web services composition are gaining a lot of interest in the area of Web services access control enforcement. There is a lot of work in this area, for instance, Srivatsa et al. [129] and Cheikh et al. in [47]. In [47] they present automatic Web services composition in trust-aware communities. This work models Web services by state transition graph and assigns explicitly the access control policy on the transition of the graph. This assignment helps them in performing the composition without defect the AC. Our work uses the idea of assigning the access control policy on the business protocol of the service but for the purpose of checking compatibility and replaceability. In [129] they present access control model and techniques for specifying and enforcing access control rules in Web service compositions.

Paci et al. [107] present an approach to determine at the design time whether a chorography can be implemented by a set of services based on their access control policies and the disclosed policies regulating the release of their credentials.

Paurobally and Jennings [111] combined the two Web service languages, WS-Conversation (WSCL) [13] and WS-Agreement. This combination is done after extending the structure of the WS-Agreement and the structure of the WS-Conversation languages (i.e., The WS-Agreement is extended to include the sender and the recipients of messages for the specification of speech-acts and WS-Conversation languages are extended to include states, sub-states, transitions and WSCL processes. As a result, non-trivial conversations in which several messages have to be exchanged before the service is completed can be specified and protocols of realistic expressiveness (such as the Contract Net protocol (Smith [127]) can be specified in our WSCL/WS-Agreement extended language.

All these models assume that a Web service provides just a single operation or that all operations are independent. Business process workflow, Web service composition, trust negation between Web services, and multi-party protocols are out of the scope of this work. We are very optimistic to integrate our work in these research trends in the near future.

Chapter 6 .WEB SERVICES ANALYSIS

This chapter discusses in details the interoperability analysis between Web services. It starts by explaining the different definitions of compatibility and replaceability between services in details. These analyses use the Web services business protocols for presenting the behavior of the services. The next parts of this chapter provide the Web services analysis after including the ACP and time constraints.

6.1. Compatibility and replaceability definitions

Formal analysis of service protocols in terms of automated support to service interoperability at the business protocol level has been discussed in some recent works (e.g. Benatallah et al. [18, 17], Ponge et al. [114], Bordeaux et al. [34], Hull et al. [82], Bultan et al. [41], Wombacher et al. [141], and Ponnekanti and Fox [115]). Compatibility and replaceability checking is one of the important interoperability analysis. There are a set of definition for compatibility and replaceability between services.

Bordeaux et al. [34] present three different definitions for compatibility and two definitions for replaceability. For compatibility, the first definition says that: *two services A and B are compatible if they have opposite behaviours (i.e., A is equivalent to* $\overline{B}$. This means that for any reachable pair of states (*s1, s2*) we have: *emissions1(s1) = receptions2(s2)* and *emissions2(s2) = receptions1(s1))* (where *emissions1*(*s1*) represents the outgoing messages from the state *s1* and *receptions1*(*s1*) represents the incoming messages to *s1*). There are some cases where two Web services may be able to cooperate in a satisfactorily way even when one has slots for receptions which the other one does not intend to use (Yellin and Strom [143], Brand and Zafiropulo [39]). As a result, the second definition based on unspecified receptions states that: *Two Web services are compatible if they have no unspecified reception*. *(i.e.,* if, for any reachable pair of states *(s1, s2),* we have that*: emis-*

sions1(s1) ⊆ receptions2(s2) and *emissions2(s2) ⊆ receptions1(s1)*). But, There is

a drawback of the two previous definitions, they do not check whether the interaction will reach a final state or not. For instance, these definitions consider that two services which do not send any message and just do receptions are compatible. To overcome this drawback, a new definition based on checking the reachability to final state in the interaction trace by checking the deadlock is emerged. The deadlock is first defined as: *A reachable pair of states (s1, s2) is a deadlock if it is impossible from these states to reach a final state*. The interaction between two services is deadlock-free if no reachable state is a deadlock. The third definition of compatibility says that*: Two services are compatible if the initial state is not a deadlock*, i.e., if there is at least one execution leading to a pair of final states.

A model for business protocols and a framework for protocol-based analysis had been presented by Benatallah et al. [19, 20, 18] and Baina et al. [12]. They studied the compatibility and replaceability issues. This model captures all of the conversations that are supported by a service. The model uses state chart and timed automata which are suitable models for describing behaviours.

They provide two definitions of compatibility, full compatibility and Partial compatibility. In the **full compatibility**: A service presented by protocol *P1* is fully compatible with another service presented by protocol *P2* if all the executions of *P1* can interoperate with *P2* □ that is, *P2* can understand any conversation that P1 can generate. In the **partial compatibility:** A service presented by protocol *P1* is partially compatible with another service presented by protocol *P2* if some executions of *P1* can interoperate with *P2*□ that is, if at least one possible conversation can take place among a service supporting *P1* and one supporting *P2*.

The drawback of the full compatibility definition is that if the service presented by protocol *P1* is compatible with the service presented by protocol *P2* then *P2* can accept all the conversation from *P1* and this means that *P2* is not necessary to accept all the conversation from *P1* (i.e. the inverse is not true). This can result an error during the interaction if the service presented by protocol *P2* tries to send a message and the service presented by the protocol *P1* is not ready to receive it.

Example 6-1: Figure 6-1 shows an example of two Web service business protocol *P1* and *P2* [21]. According to the definition of compatibility defined by Benatallah et al. [19, 20, 18], the two services supporting the protocols *P1* and *P2* in Figure 6-1 are fully compatible (i.e., all the executions of *P1* can interoperate with *P2*). Also, in Figure 6-2 the two services 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 6-2 shows that the Web service supporting the protocol *P2* has not the ability to receive the **CancelOrder(-)** message from the Web service supporting *P1*. As a result, there is a potential conversation which will not be accomplished and results in an error.

Therefore, in our work we will adapt a new definition of compatibility based on the error free interaction. We merge the second and the third definition of Bordeaux et al. [34]. In other words, two services are compatible if and only if any potential send message from one service can be received by the other service during their interaction and vice versa and there is no deadlock. Based on our definition, if two services are compatible, we guarantee that no error can be happen during the interaction. As a result, the Web service supporting the two protocol of Figure 6-2 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 the Web service supporting *P2* cannot receive this message.

In terms of timed business protocol; Ponge et al. [114] handle timed protocols and built their analysis on the definition of compatibility (full compatibility and partial compatibility) of Benatallah et al. [19, 20, 18]. They formalized the C-Invoke constraints which define time windows of availability and M-Invoke constraints which define expirations deadlines. Also, they converted the timed business protocol to timed automata and defined a set of operators like composition, intersection, and projection. For instance, to check full compatibility between two Web service supporting protocol *P1* and *P2*, if the projection by *P1* of the composition between *P1* and *P2* is equal to *P1*, then the Web service supporting *P1* and the Web service supporting *P2* are compatible (i.e. if [*P1 composition P2*] *projection P1= P1*, then the Web service supporting *P1* is fully compatible with the Web service supporting *P2*).

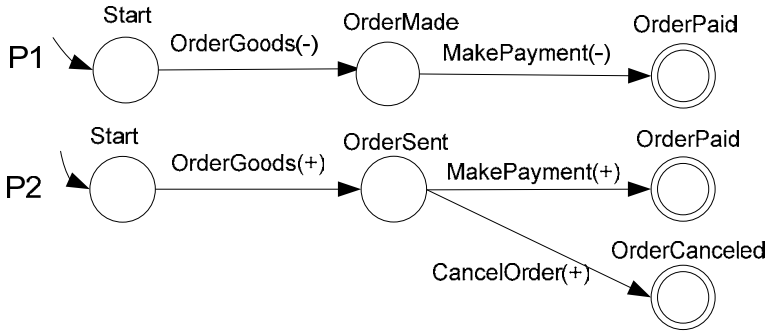

Figure 6-1: Two business protocols of compatible Web services (full compatibility).

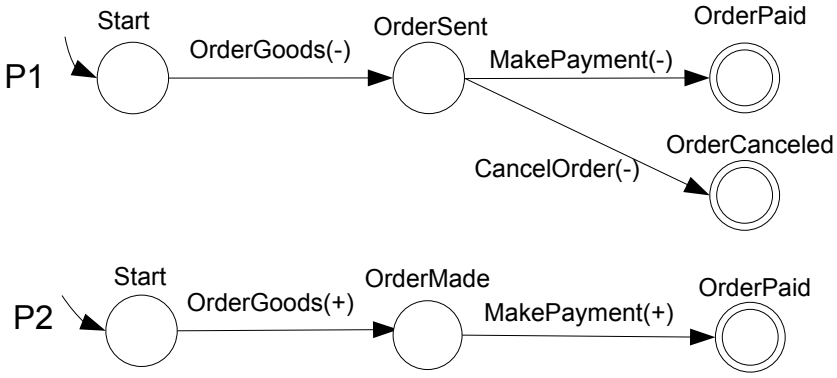

Figure 6-2: Two business protocols of incompatible Web services.

In replaceability analysis, Bordeaux et al. [34] present two definition of replaceability (or substitutability). The first definition is context-dependent and state that: *In a particular application made of two compatible services A and B, service A can replace service A if A is also compatible with B*. The second definition is a context-independent and state that: *A service A can replace a service A if it is compatible with any service B which is compatible with A*. Benatallah et al. [19, 20, 18] defined set of classes of replaceability (e.g., protocol equivalence w.r.t. replaceability; protocol subsumption w.r.t. replaceability; protocol replaceability w.r.t. a client protocol; and protocol replaceability w.r.t. an interaction role).

For replaceability, we will use the definitions of Bordeaux et al. [34] on our work because they guarantee the error interaction free after replaceability due to its dependency on compatibility.

The rest of this chapter is divided into two main parts. The first part discusses the compatibility and replaceability after assigning the time and AC constraints. The second part presents the interoperability analysis between Web services after assigning message specifications constraints.

6.2. Compatibility and replaceability analysis after assigning time and AC

In this section, we will start by defining the interaction traces between Web services and compatibility between them in terms of interaction traces. Then, the compatibility and the replaceability in terms of the product automata and in terms of the intersection automata respectively are defined. We will present a set of formal definitions leading to the definition of compatibility and replaceability, such as the definition which is used to construct the product automata, intersection automata, and the definition which is used for checking the co-accessibility of a state in automata. We will prove that the compatibility in terms of interaction traces is equivalent to the compatibility in terms of product automata (see proof 6-1 and proof 6-2 in the end of this chapter).

Firstly, we will recall our definition of the timed business protocol with AC. As shown in the previous chapters, the one-clock time business protocol is a special case of the multi-clocks timed business protocol. Therefore, the analysis of the multi-clocks timed business can be applied on one-clock timed business protocols. The protocol which is used in the analysis is the multi-clocks protocol after removing the implicit transitions. We will call it simply timed business protocol with ACP.

Definition 6-1 (Time business protocol with AC) *A time business protocol with AC is a 5-tuple P = (S; s_0; X; T; F) which consists of the following elements:*

S is a finite set of states.

□ $s_0 \in S$, *is the initial state.*

□ *X is a finite set of clocks with a set of clock reset* $CR \subseteq X$.

□ $T \subseteq S^2 \times M \times (\{-\} \times 2^c) \cup (\{+\} \times PL)) \times Tc \times CR \times RK$, *is a finite set of explicit transition where M is set of input/output messages,* $\{+, -\}$ *polarity of the message where {+} means input message and {□} means output message. PL is the set of access control policies, c is the set of credentials Tc is the time constraint which is a disjunction of the conjunction of static and dynamic atomic time constraints and CR is the set of clock reset. RK is the set of variables reset, if there is no dynamic variable in the protocol then this set is always empty.*

□ *This protocol is deterministic (i.e., all the output transitions from any state are different and there is no overlapping between the output messages).*

□ $F \subseteq S$ *is a set of final states. If* $F = \emptyset$ *then P is said to be an empty protocol.*

□ *All states in the automata are accessible and co-accessible.*

6.2.1. Compatibility

In the following sections, we will refer to the Web service by its business protocol. Therefore, when we say protocol *P1* is compatible with protocol *P2*, we mean that

the Web service supporting the protocol *P1* is compatible with the Web service supporting the protocol *P2*.

Checking the compatibility and replaceability between explicitly timed business protocols is a straightforward operation. Our compatibility checking definition is based on the comparison between constraints on the explicit transitions. The time constraints obtained after the conversion process can be static, the constraint time

interval values are fix (e.g., $x1 \in [0, 2[$), or dynamic in such case the time interval is

changeable (e.g., $x1 \in [0,k[$) where k obeys to the sequence {1,2,4,□ .}. The types

of the time constraints that are used in the comparison operation can be two static, two dynamic, or one static and the other one is dynamic. For comparing two static constraints, the inclusion between the two time intervals can be determined by comparing the interval limits. For comparing a static time constraint with a dynamic time constraint, we check that for all values of the variable k, the static time interval is included or not in one resulted static time intervals after substituting *k* by its values. For comparing two dynamic constraint, we check the set of intersection between the static time intervals of the dynamic time intervals of the two constraint

for each value of the used variable (e.g., the variable *k* in the previous example $x1 \in$

$[0, k[$).

Definition 6-2 (Intersection between time intervals) *the intersection between two time intervals I_1 and I_2 is denoted by $(I_1 \cap I_2)$ where:*

- *If $I_1=[a1,b1]$and $I_2=[a2,b2]$ are static time interval then $(I_1 \cap I_2) =[a1,b1]$ if $a1>=a2$ and $b2>=b1$, or $(I_1 \cap I_2) =[a2,b2]$ if $a2>=a1$ and $b1>=b2$, or($I_1 \cap I_2$)*

=Null if (a2>a1 and b1<a2) or (a1>a2 and b2<a1), or $(I_1 \cap I_2)$ =[a2,b1] if a2>=a1 and b2>=b1 , or $(I_1 \cap I_2)$ =[a1,b2] if a2<=a1 and b2<=b1.

- *If I_1=[a1,b1] is a static time interval and I_2=[a+k*b, c+k*d[with k={0,1,2,□} is a dynamic time interval, where the dynamic time interval is a set of static time intervals after the substitution of the variable k by its values, the intersection $(I_1 \cap I_2)$ is the set of intersection between the time interval I_1 and each element of the dynamic set where each element is calculated by the intersection between the time interval I_1 and one of the static time intervals of the dynamic time interval I_2 which is calculated using the previous item.*
- *If I_1=[a1+k*b1,c1+k*d1[with k={0,1,2,□} is a dynamic time interval and I_2=[a2+k*b2,c2+k*d2[with k={0,1,2,□} is a dynamic time interval, the intersection $(I_1 \cap I_2)$ is the set of intersection between the static time intervals of the dynamic time interval I_1 and the static time intervals of the dynamic time interval I_2, where each element on the intersection set is calculated by the intersection of the two static time interval on I_1 and I_1 having the same value of the variable k.*

Example 6-2:

- If I_1=[3,6]and I_2=[2,5] two static time intervals, then $I_1 \cap I_2$=[3,5]
- If I_1=[1,4] is a static time interval and I_2=[a+k*b, c+k*d[with k={0,1,2,□ } is a dynamic time interval with a=2 , b=3, and c=∞ (i.e., I_2={[2+0*3,∞[, [2+1*3,∞[,[2+2*3,∞[, ... }, then $I_1 \cap I_2$={[2,4]}
- If I_1=[a1+k*b1,c1+k*d1[with k={0,1,2,□ } is a dynamic time interval with a1=1 , b1=1, and c1=∞ (i.e., I_2={[1+0*1,∞[, [1+1*1,∞[,[1+2*1,∞[, ... } and I_2=[a2+k*b2,c2[with k={0,1,2,□ } is a dynamic time interval with a2=2 , b2=3, and c2=∞ (i.e., I_2={[2+0*3,∞[, [2+1*3,∞[,[2+2*3,∞[, ... }), then $I_1 \cap I_2$={[2,∞[, [5,∞[,[7,∞[, ... }

Definition 6-3: An interaction trace *IT between a protocol $P^1 = (S^1; s_0^1; X^1; T^1; F^1)$ and a protocol $P^2 = (S^2; s_0^2; X^2; T^2; F^2)$ is a finite sequence $((s_i^1; s_i^2 ; \overline{m_i} ; c_i; pl_i ; t_i; s_{i+1}^1; s_{i+1}^2))_i$, where $s_n^1 \in S^1$ and $s_n^2 \in S^2$, m_i is a message instance with its direction (either ← or →). c_i is the set of all credentials sent either in this message or*

in any previous message with the same direction and pl_i is the ACP. t_i is the time period since the previous message, or zero for the first message.

***Definition 6-4**: **An interaction trace** IT = $((s_i^1; s_i^2 ; \overrightarrow{m_i} ; c_i; pl_i; t_i; s_{i+1}^1; s_{i+1}^2))_i$ between a protocol $P^1 = (S^1; s_0^1; X^1;T^1; F^1)$ and protocol $P^2 = (S^2; s_0^2;X^2; T^2; F^2)$ is **correct** if and only if for every tuple $(s^1; s^2; \overrightarrow{m} ; c; pl; t; s^1; s^2)$ in IT (and symmetrically for every tuple $(s^1; s^2; \overrightarrow{m} ; c; pl; t; s^1; s^2)$:*

- *There are two transitions $(s^1; s^1; M^-; C; tc^1) \in T^1$ and $(s^2; s^2; M^+; P; tc^2) \in T^2$; where m is an instance of M and tc^1 and tc^2 are the time constraint.*
- *The set of credentials instances c sent in the instance message m match the access control policy pl;*
- *$[tc^1]_{CV(t)\,(x^1)}$ =true and $[tc^2]_{CV(t)\,(x^2)}$ =true for all clocks $x^1 \in X^1$ and $x^2 \in X^2$.*
- *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 transition $(s_{n-1}^1 ; s_{n-1}^2 ; \overrightarrow{m_{n-1}} ; c_{i-1}; pl_{i-1};t_{i-1}; s_n^1; s_n^2)$, $s_n^1 \in F^1$ and $s^2{}_n \in F^2$*

***Definition 6-5**: **(Compatibility in terms of interaction trace assigned with AC)** Two business protocols, $P^1 = (S^1; s_0^1; X^1;T^1; F^1)$ and a protocol $P^2 = (S^2; s_0^2; X^2;T^2;$*

F^2) are compatibles in terms of interaction trace if $\forall tr \in IT(P^1; P^2)$, its last tuple

$(s_{n-1}^1; s_{n-1}^2; \overrightarrow{m_{n-1}}; c_{n-1}; pl_{n-1}; t_{n-1}; s_n^1; s_n^2)$ verifies:

$\forall m \; \forall t$ if $\exists (s_n^1; s_{n+1}^1; M_n^-; c_n^1; tc_n^1) \in T^1$, such that $[tc_n^1]_{CV(t)\,(x1)}$ =true and m is

an instance of M_n then $\exists \; s^2 \in S^2$ such that $tr.(s_n^1, s_n^2, \overrightarrow{m}; c_n; pl_n; t; s_{n+1}^1; s_{n+1}^2) \in$

$IT(P^1; P^2)$, and the set of credentials instances c_n sent in the message m satisfies the policy pl_n..

$\forall m \; \forall t$ if $\exists (s_n^2; s_{n+1}^2; M_n^-; c_n^2; tc_n^2) \in T^2$, such that $[tc_n^2]_{CV(t)\,(x^2)}$ =true and m is

an instance of M_n then $\exists s^1 \in S^1$ such that $tr.(s_n^1, s_n^2, \overrightarrow{m}, c_n; pl_n; t; s_{n+1}^1; s_{n+1}^2) \in$

$IT(P^1; P^2)$, and the set of credentials instances c_n sent in the message m satisfies the policy pl_n.

6.2.1.1. Cumulative access control credentials

There is a difference in the methodology between checking the compatibility in terms of message exchange and in terms of AC. Checking the compatibility in terms of message exchange depends on the current message of each protocol and

corresponding current message in the other protocol. But checking the compatibility in terms of AC depends on the current ACP and the current or the previous 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 credentials cumulative access control credentials.

Figure 6-3 shows an example of two business protocols *P1* and *P2* where *P1* provide its **Visa** credential in the first transition but *P2* asked for it in the third transition. If we compare the two protocols without calculating the cumulative ACC (CACC) then we will find that they are incompatible but after calculating the CACC for *P1* we find that they are compatible. The question is then; in which step in the checking process the satisfaction between ACP and credentials can be checked? Figure 6-4 shows an example to answer this question. In this figure, *P1* is a client protocol and *P2* is a service protocol and *P1* is compatible with *P2*. *P3* is another service protocol which is not compatible with *P1* because the policy (**Visa**, **Student-Card**) in *M7* in *P3* can not satisfied by ((**Visa,Student-Card)** ∨ (**Visa**,**Prof-Card**)) credentials in *P1*.

Figure 6-5 shows an example where protocol *P2* has a policy (**Visa**, **Student-Card**) not satisfied by ((**Visa**,**Student-Card)** ∨ (**Visa, Prof-Card)**) credentials in *P1* but the two protocols are compatible because there is a part of the credentials (**Visa,Prof-Card**) will not provided by the protocol *P1* if it interacts with *P2* and the provided credentials in *M7* is (**Visa**, **Student-Card**) which satisfy the (**Visa**, **Student-Card**) in *P2*. Therefore, we must not calculate the cumulative before determining the transition which will be used in the interaction between the two protocols. Thus, when we update the CACC, we only consider the transition which will potentially share in the interaction.

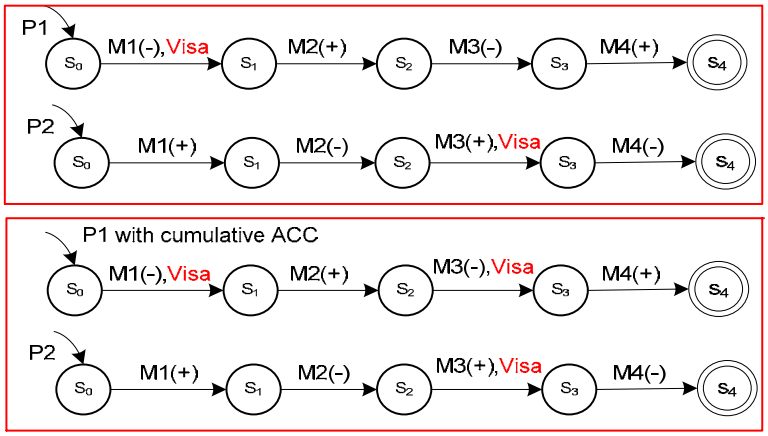

Figure 6-3: Importance of calculating the cumulative access control credentials.

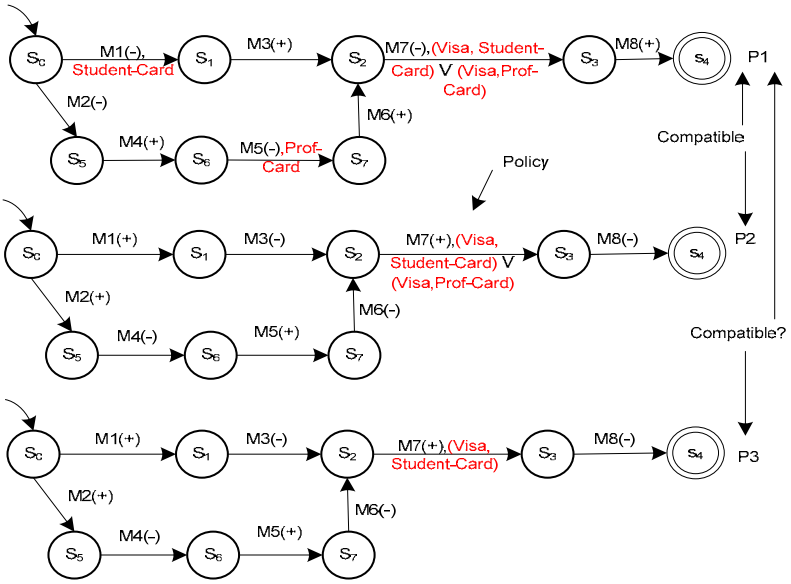

Figure 6-4: Three different business protocols *P1, P2*, and *P3*. *P1* is compatible with *P2* but not compatible with *P3*.

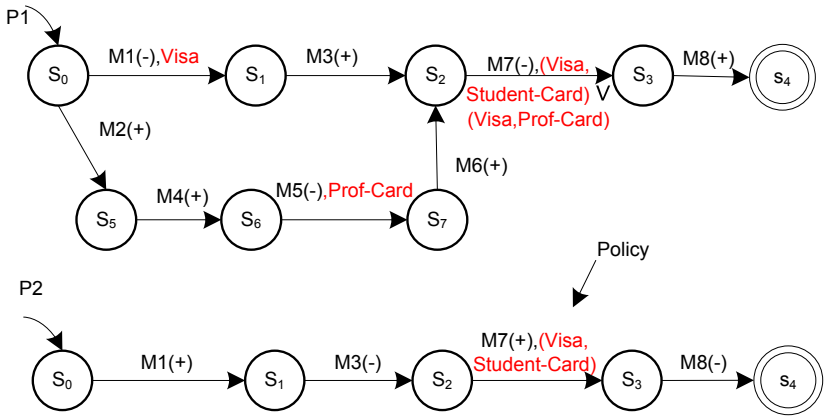

Figure 6-5: *P1* is compatible with *P2* (this show the importance of calculating the cumulative *ACC* after determining the transition which will be used in the interaction and this is accomplished by calculating the product automata).

Definition 6-6 (Product automata) *The product automata* A^p *of two timed explicitly timed business protocols* $P^1 = (S^1; s_0^1; X^1; T^1; F^1)$ *and a protocol* $P^2 = (S^2; s_0^2; X^2; T^2; F^2)$ *) is defined as* $A^p = (S^p; s_0^p; T^p; X^p; F^p)$ *where:*

- $S^p = S^1 \times S^2$
- $s^p = (s_0^1 ; s_0^2)$
- T^p *is the greatest subset of* $((S^1 \times S^2) (S^1 \times S^2) \times ((\overline{M} \times 2^{c1} \times pl^2) \vee (\overline{M} \times 2^{c2} \times pl^1)) \times X \times tc^P)$ *such that for all transition* $((s_i^1; s_i^2) ; (s_{i+1}^1; s_{i+1}^2); \overline{m_i}; pl^p ; c^p; tc_i^p) \in T^p$ *there exist two transitions* $(s_i^1; s_{i+1}^1; m_i ;(pl^1 \text{ or } c^1) ; tc_i^1) \in T^1$ *and* $(s_i^2; s_{i+1}^2; m_i ;(pl^2 \text{ or } c^2) ; tc_i^2) \in T^2$ *with:*

- $tc_i^p=(tc_i^1, tc_i^2)$
- $polarity(m_i; P^1) \neq polarity(m_i; P^2)$ and
  - If $polarity(m_i; P^1) = -$ then $\overline{m_i} = \overline{M}$, $pl_i^p = pl_i^2$ and $c_i^p = c_i^1$
  - Otherwise $(m_i; P^2) = -$ then $\overline{m_i} = \overline{M}$, $pl_i^p = pl_i^1$ and $c_i^p = c_i^2$

- $X_i^{p=} \{X_i^1, X_i^2\}$
- $F^p = F^1 \times F^2$

6.2.1.2. Clocks synchronization

In multi-clocks Web services, comparing two constraints from two different protocols during the compatibility and replaceability checking must be performed after clock synchronization process. In this process, the relations between the clocks in each protocol are determined on each transition on the product automata. It uses the same technique as for calculating the cumulative access control credential.

Definition 6-7 *(Clocks synchronization constraint): a clock synchronization constraint can be in the form:*

- $x_i = x_j + a$, *where* $x_i, x_j \in X$ *and* $a \in R$.

- $x_i < x_j + a$.

Definition 6-8 *(Clocks synchronization in the product automata): Given* A^p *of two protocols transition* $P^1 = (S^1; s_0^1; X^1; T^1; F^1)$ *and a protocol* $P^2 = (S^2; s_0^2; X^2; T^2; F^2)$ *), for each transition* t^p *in* A^p*, let tc be time constraint and y be a clock variable, SC(tc,y) is defined as:*

- $SC(x=a,y)=\{\{y=x-a\}\}$

- $SC(x \in [a,b],y)=\{\{x-b \leq y, y \leq x-a\}\}$

- $SC(x \in [a,b[,y)=\{\{x-b < y,\ y \leq x-a\}\}$

- $SC(x \in]a,b],y)=\{\{x-b \leq y,\ y < x-a\}\}$

- $SC(x \in]a,b[,y)=\{\{x-b < y,\ y < x-a\}\}$

- $SC(tc_1 \wedge tc_2, y)=\{CSC_1 \cup CSC_2 \mid CSC_1 \in SC(tc_1,y) \text{ and } CSC_2 \in SC(tc_2,y)\}$

- $SC(tc_1 \vee tc_2, y)= SC(tc_1,y) \cup SC(tc_1,y)$

Let the time constraint $tc^p=(tc^1, tc^2)$ *and clock reset* $\{x,y\}$ *are on the product transition* t^p, *then* SC^p *is defined as* $\{\{x=y\} \cup csc \mid SC(tc^1 \wedge tc_2)\}$

Definition 6-9 (Update and Merge operators) *the update operator UPDATE*

- *For two credentials sets is the union of the two set (AND operator) (i.e., C1 UPDATE C2= C1 AND C2, where C1 and C2 are credentials sets) .*

- *Let CSC be a set of synchrounous clocks constraints. We define* $CSC|_x$ *={csc | csc ∈CSC and csc is of the form x=y+a, y=x+a, x≤y+a, y≤x+a},*

$$UPDATE(SC, SC',y)=\{(CSC\backslash \bigcup_{x\in y} CSC\ |_x) \cup CSC' \mid CSC \in SC, CSC' \in SC'$$

Let t^p *be a product transition with time constraint* $tc^p=(tc^1,tc^2)$ *and clock reset {x,y},* $UPDATE(SC, t^p)=UPDATE(SC,SC^p(t^p),\{x,y\})$

- *for two credentials sets is the logical disjunction of the two set (OR operator)*

- $MERGE(SC_1, SC_1)= SC_1 \cup SC_2$

Definition 6-10. (Cumulative path in the product automata)*:* $PA^p = ((s_i^1,s_i^2) \xrightarrow{CumV_i} (s_{i+1}^1,s_{i+1}^2) ; \square ;(s_n^1,s_n^2) \xrightarrow{CumV_n} (s_{n+1}^1,s_{n+1}^2))$ *is a cumulative path in the product automata* $A^p = (S^p; s_0^p; T^p; F^p)$ *where*

□*States* $(s_i^1 ; s_i^2); \square ; (s_n^1; s_n^2) \in S^p$.

- *Each cumulative value* $CumV_i$ *is the set of cumulative values which is the UP-DATE of the previous set of cumulative values* $CumV_{i-1}$ *and the current set of Value* V_i *where* V_i *is the set of values on the transition between the state* $((s_i^1, s_i^2)$ *and* $(s_{i+1}^1, s_{i+1}^2))$ *and* $CumV_0 = V_0$.
- *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 6-11. (Co-accessibility of a state in the product of automata) $A^p = P^1 x P^2 = (S^p; s_0^p; T^p; F^p)$ *is the product of automata of two TBP,* P^1 *and* P^2*, state* $(s_i^1; s_i^2) \in Sp$ *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 6-12. (Compatibility in terms of product automata assigned with AC) *Protocols* $P^1 = (S^1; s_0^1; T^1; X^1; F^1)$*, protocol* $P^2 = (S^2; s_0^2; T^2; X^2; F^2)$*, and* $A^p = (S^p; s_0^p; T^p; F^p)$ *is their product automata, we say that* P^1 *and* P^2 *are compatible using their product automata if there is a relation* $R \subseteq S1 \times S2$ *where for all* $(s_i^1, s_i^2) \in R$:

$\forall(s_i^1; s_{i+1}^1 ; m^-; c_i^1; tc_i^1) \in T^1, \exists (s_i^2; s_{i+1}^2 ; m^+; pl_i^2; tc_i^2) \in T^2$, *where* $(s_i^1; s_i^2 ; s_{i+1}^1 ; s_{i+1}^2 ; \overline{M}; c_i^1$ *and* $pl_i^2 ; tc_i^p) \in T^p$ *and* $(s_{i+1}^1, s_{i+1}^2) \in R$.

$\forall(s_i^2; s_{i+1}^2 ; m^-; c_i^2; x_i^2; tc_i^2) \in T^2, \exists (s_i^1; s_{i+1}^1 ; m^+; pl_i^1; x_i^1; tc_i^1) \in T^1$, *where* $(s_i^1; s_i^2 ; s_{i+1}^1 ; s_{i+1}^2 ; \overline{M}; c_i^2$ *and* $pl_i^1 ; tc_i^p) \in T^p$ *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 $PA^p = ((s_0^1, s_0^2) \xrightarrow{CumV_0} (s_1^1, s_1^2)$; ; $(s_n^1, s_n^2) \xrightarrow{CumV_n} (s_{n+1}^1, s_{n+1}^2))$, *in the product automata, in each transition, each policy is satisfied by the set of cumulative credential values on this transition and the inter-*

section between the time constraints and the cumulative time constraint value is equal to the time constraint of the sending transition.

The previous definition is used as a base for the compatibility checking algorithm between two protocols in terms of product automata with time constraints and AC. The algorithm which is used for checking the compatibility between two protocols in terms of product automata with time and AC can be divided into two parts. The first part is for checking compatibility in terms of message exchange. This algorithm can be implemented 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 **is 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 and the time constraints satisfied), 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. This is shown in Algorithm 6-1. The second part is for calculating the cumulative credentials and time constraint on each transition on the product automata.

Algorithm 6-2 presents the second part of the algorithm. The idea of this algorithm is to use the queue data structure to cumulate the credentials and the time constraints. Each element of the queue consists of the state, cumulative value (credentials or time) corresponding the protocol *P1* in this state, and the cumulative value corresponding the protocol *P2* in this state. The algorithm traverses through the automata for updating these values of the states and in the same time updates the cumulative value on the transitions. After calculating the cumulative values 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 or the time intervals intersection is not equal to the time intervals of the sending messages, then the two protocols are not compatible in terms of AC or time.

Algorithm 6-1: Compatibility between two protocols using product automata in terms of message exchange

Input: $P1 = (S^1; s_0^1; T^1; F^1)$ and $P2 = (S^2; s_0^2; T^2; F^2)$ and their product automata $A^p = (S^p; s_0^p;$

$T^p; F^p)$

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.

$C_{message}$ // boolean variable set to true if the two protocols are compatible in terms of message exchange and false otherwise.

modifiedEca ← true // Boolean variable used for verifying the co-accessibility, it is true each time E_{ca} changed.

E_c ← $(s_0^1; s_0^2)$

//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

$\forall(s_i^1; s_{i+1}^1 ; m^-; tc_i^1) \in T^1$

If $\exists$ $(s_i^2; s_{i+1}^2 ; m^+; tc_i^2) \in T^2$, where $(s_i^1; s_i^2 ; s_{i+1}^1 ; s_{i+1}^2 ; \overline{M}; tc_i^1) \in T^p$ and $(s_{i+1}^1, s_{i+1}^2) \in R$

then

Continue

Else

Return false

End

//verifying the output message from *P2*

$\forall(s_i^2; s_{i+1}^2 ; m^-; tc_i^2) \in T^2$

If $\exists$ $(s_i^1; s_{i+1}^1 ; m^+; tc_i^1) \in T^1$, where $(s_i^1; s_i^2 ; s_{i+1}^1 ; s_{i+1}^2 ;$ M ; $tc_i^2) \in T^p$ 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 $(s_i^1 ; s_i^2) \in E_c \notin E_{ca}$ **do**

If $\exists(s_j^1 ; s_j^2) \in E_{ca}$ and $((s_j^1 ; s_j^2); (s_i^1 ; s_i^2); m_i; tc^1_i) \in T^p$ then

$E_{ca} \leftarrow E_{ca} \cup (s_i^1 ; s_i^2)$ and *modifiedEca* ← true

End

End

End

If $E_c - E_{ca} \neq 0$ then

Return false

Else

$C_{message}$ =True (the protocols are compatible in terms of messages changes and time)

End

Complexity analysis: Let *T1* and *T2* be the number of transitions of the two protocols *P1* and *P2* respectively, the construction of the product automata will take (*T1* x *T2*). As a result, the complexity for the algorithm will be O(*T1* x *T2*).

Algorithm 6-2: Compatibility between two protocols using cumulative product automata in terms of AC and time.

Input: P1 = (S^1; s_0^1; T^1; F^1) and P2 = (S^2; s_0^2; T^2; F^2), product automata A^p = (S^p; s_0^p; T^p; F^p)

Output: protocols P1 and P2 are compatible in terms of AC or not.

1- Calculate the cumulative values on the automata.

v_i^1 : Cumulative value corresponding to protocol P^1 and assigned to the state s_i.

v_i^2 : Cumulative values corresponding to protocol P^2 and assigned to the state s_i.

v_{ij}^1 : Cumulative values corresponding to protocol P^1 and assigned to the transition between s_i and s_j (i.e. union of set of values in those transitions).

v_{ij}^2 : Cumulative values corresponding to protocol P^2 and assigned to the transition between s_i and s_j .

p_i^1 : The policy on the transition t_i in the protocol P1.

c_i^1 : The cumulative credentials of the protocols P1 on the transition t_i on the product automata

tc_i^1: the cumulative time constraint of the protocols P1 on transition t_i in the product automata.

for each state $s_i \in output(s_0)$ **do**

$v_i^1 = v_{0i}^1$

$v_i^2 = v_{0i}^2$

ENQUEUE(s_i; v_i^1 ; v_i^2)

while Q ≠ empty do

Temp_Q = DEQUEUE(Q)

foreach $s_j \in output(s_i)$ in (s_i; v_i^1 ; v_i^2) = Temp_Q **do**

v_j^1 _ temp = v_j^1

v_j^2 _temp = v_j^2

if $v_j^1 \neq$ null then

$v_j^1 = (v_{1j}]^{\uparrow}1$ UPDATE v_i^1) MERGE c_j^1

else

$v_j^1 = (v_{1j}]^{\uparrow}1$ UPDATE v_i^1)

if $v_j^2 \neq$ null then

$v_j^2 = (v_{1j}]^{\uparrow}2$ UPDATE v_i^2) MERGE v_j^2

else

$v_j^2 = (v_{1j}]^{\uparrow}2$ UPDATE v_i^2)

$v_{ij}^1 = v_{ij}^1$ UPDATE v_i^1

$v_{ij}^2 = v_{ij}^2$ UPDATE v_i^2

if $\neg$ ((v_j^1 == v_{j^1} temp) and $v_i^1 \neq$ null and (v_j^2 == v_{j^2} temp) and $v_j^2 \neq$ null)) then

ENQUEUE(Q; s_j ; v_j^1 ; v_j^2)

2- if $\forall p_i^1$, $c_i^1 \subseteq p_i^1$ satisfied and ($tc_i^1 \cap tc_i^2 = tc_i^1$) , and $\forall p_i^2$, $c_i^2 \subseteq p_i^2$ satisfied and

($tc_i^1 \cap tc_i^2 = tc_i^2$) where $c_i^1 = v_i^1$ which is calculated in step 1 for credentials, $c_i^2 = v_i^2$(credentials), $tc_i^1 = v_i^1$ which is calculated in step 1 for time, $tc_i^2 = v_i^2$(time)

then

The two protocols are compatible in terms of AC and time.

Else

The two protocols are not compatible in terms of AC or time.

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* x *T2*). The calculation of the cumulative credentials will take number of states in the product automata |(*S1* x *S2*)| multiplied by the size of the longest non looping path multiplied by |(*S1* x *S2*)| (i.e. cumulative credentials takes $|(S1 \times S2)|^3$) . As a result, the complexity for the algorithm will be $((T1 \times T2) + |(S1 \times S2)|^3)$.

Example 6-3: In this example, a business protocol of a web service performing two operations with two different AC and a client service that interacts with this service are presented in Figure 6-6 and Figure 6-7 respectively. Figure 6-8 and Figure 6-9 show the product automata of the two protocols and the graphical representation of the used ontology. Note that the compatibility of the two protocols partially depends on the subsumption relation between school and student cards. Using the ontology, the two protocols are compatible because the **(School student card)** is a (**student card)**.

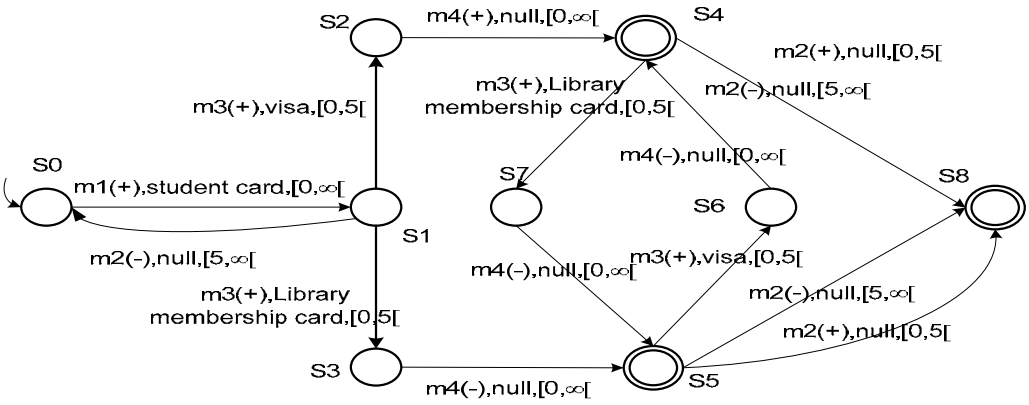

Figure 6-6: Business protocol of web service performs two operations.

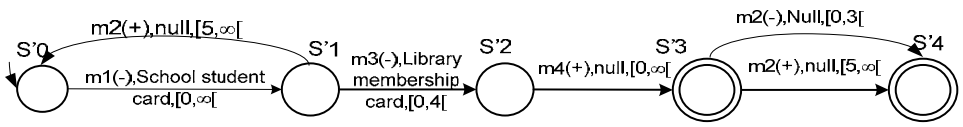

Figure 6-7: Business protocol of a consumer needs to interact with the service in Figure 6-6.

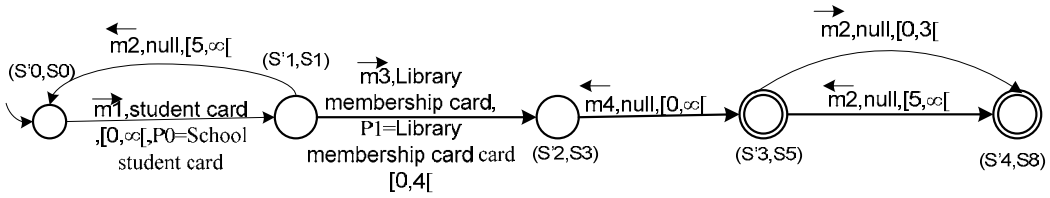

Figure 6-8: Product automata of the two protocols of Figure 6-6 and Figure 6-7 assigned with AC.

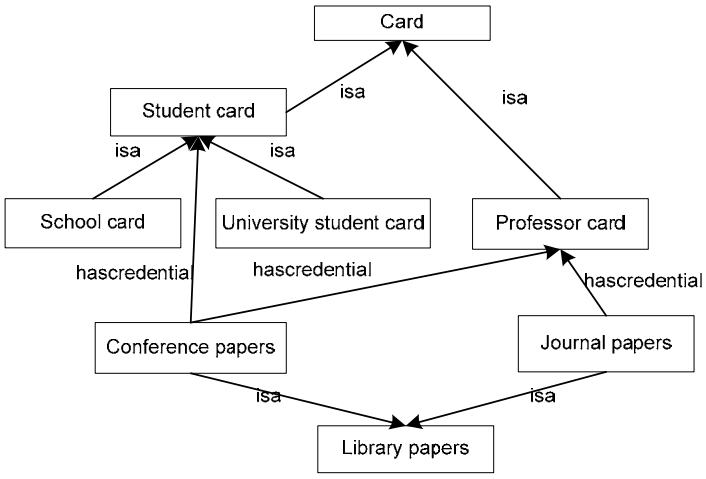

Figure 6-9: Graphical representation of resources ontology linked with the credential ontology.

6.2.2 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 6-13 (Intersection automata) *The intersection automata A^i of two timed business protocols with AC, $P^1 = (S^1; s_0^1; X^1; T^1; F^1)$ and $P^2 = (S^2; s_0^2; X^2; T^2; F^2)$) is defined as $A^i = (S^i; s_0^i; T^i; X^p; F^i)$ where:*

- $S^i = S^1 \times S^2$
- $s^i = (s_0^1 ; s_0^2)$
- T^i is the greatest subset of $((S^1 \times S^2) (S^1 \times S^2) \times ((M^- \times 2^{c1}) \vee (M^+ \times pl^1)) \times X \times$

I) such that for all transition $((s_i^1; s_i^2) ; (s_{i+1}^1; s_{i+1}^2))$;  $; pl^i; c^i; tc_i^i) \in T^i$ *there exist two transitions* $(s_i^1; s_{i+1}^1; m_i;(pl^1 \text{ or } c^1) ; x_i^1 ; tc_i^1) \in T^1$ *and* $(s_i^2; s_{i+1}^2; m_i;(pl^2 \text{ or } c^2) ; x_i^2 ; tc_i^2) \in T^2$ *with :*

  - $tc_i^i = (tc_i^1, tc_i^2)$
- $polarity(m_i; P^1) = polarity(m_i; P^2)$ *and*
  - *If* $polarity(m_i; P^{1,2}) = -$ *then* $c_i^i = (c_i^1 \text{ and } c_i^2)$
  - *otherwise* $(m_i; P^{1,2}) = +$ *then* $pl_i^i = (pl_i^1 \text{ and } pl_i^2)$
- $F^i = F^1 \times F^2$

Definition 6-14 (Replaceability in terms of intersection automata) *Protocol* $P^1 = (S^1; s_0^1; T^1; X^1; F^1)$, *protocol* $P^2 = (S^2; s_0^2; T^2; X^2; F^2)$, *and* $A^i = P^1 \cap P^2 = (S^i; s_0^i; T^i; F^i)$ *is their intersection automata, we say that* P^1 *can be fully replaced by* P^2 *using*

their intersection automata if there is a relation R ⊆ S1 x S2 where for all (s_i^1, s_i^2)

$\in R$:

- $\forall(s_i^1; s_{i+1}^1; m_i^+; pl_i^1; tc_i^1) \in T^1 \quad \exists (s_i^2; s_{i+1}^2; m_i^+; pl_i^2; tc_i^2) \in T^2$ *,where* $(s_i^1; s_i^2; s_{i+1}^1; s_{i+1}^2; \overrightarrow{M}; pl_i^i; tc_i^i) \in T^i$ *, and* $.(s_{i+1}^1, s_{i+1}^2) \in R$

- $\forall(s_i^2; s_{i+1}^2; m_i^-; c_i^2; i_i^2) \in T^2 \quad \exists (s_i^1; s_{i+1}^1; m_i^-; c_i^1; i_i^1) \in T^1$ *, where* $(s_i^1; s_i^2; s_{i+1}^1; s_{i+1}^2; \overrightarrow{M}; c_i^i; i_i^2) \in T^i$*, 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 product automata from this state to final state.*

$\square(s_0^1, s_0^2) \in R$

$\square$*For all the complete cumulative paths* $PA^i = ((s_0^1, s_0^2) \xrightarrow{[\ldots]} (s_{i+1}^1, s_{i+1}^2)$; $\square$ $;(s_n^1, s_n^2) \xrightarrow{[\ldots]} (s_{n+1}^1, s_{n+1}^2))$, *in the intersection automata, any set of credential satisfy the cumulative policy* Pl_i^2 *can also satisfy the cumulative policy* Pl_i^1 *and* $c_i^1 \subseteq C_i^2$.

The algorithm for checking the replaceability uses the same mechanism which is used in the compatibility algorithm. The idea is to traverse through the intersection automata starting from the initial state and checking the mentioned properties of the relation *R*. The second part uses the same technique as algorithm 2 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 Pl_i^2 can also satisfy the cumulative policy Pl_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.

Proof 6-1: Compatibility in terms of automata implies compatibility in terms of interaction trace.

Given $P(S,s_0,T,X,F)$ and $P\square(S\square,s\square_0,T\square,X\square,F\square)$ are two TBP, and $A^p = (S^p; s_0^p; T^p; X^p; F^p)$ is a product automata, suppose we have $tr = (S_0,S\square_0,m_1,m\square_1, t_1, S_1, S\square_1)$, $(S_1,S\square_1,m_2,m\square_2, t_2, S_2, S\square_2)$,$\square$, $(S_{i-1},S\square_{i-1},m_i,m\square_i, t_i, S_i,S\square_i)$ is a partial interaction trace. We have *P* and $P\square$ are compatible in terms of automata so there is a relation

$R \subseteq S \times S\square$ where for all $(S_i,S\square_i) \in R$:

- ∀ $m_i \in$ output(S_i), if polarity (m_i,P) = - then ∃ $I(S_i, S_{i+1}, m_i^-, tc) \in T$ and ∃ $I(S_i, S_{i+1}, m_i^+, tc) \in T$ where $C_i^- \subseteq pl_i^+$ (C_i is the set of credentials and pl is the ACP), and ∃ $I_p(S_i, S_i, \overline{m_i}, t_{i+1}, S_{i+1}, S_{i+1}) \in T_P$, and $(S_{i+1},S_{i+1}) \in R$, which implies that there is a new transition added to the partial interaction trace tr is$(S_i, S_i, \overline{m_i}, t_i, S_{i+1}, S_{i+1})$ and because (S_{i+1},S_{i+1}) is co-accessible (there is a path to the final state) then the interaction trace $IT(tr.(S_i, S_i, \overline{m_i}, t_i, S_{i+1}, S_{i+1})$ can reach a final state. Therefore $IT(tr.(S_i, S_i, \overline{m_i}, t_i, S_{i+1}, S_{i+1})$ is start by initial state and ends by final state so it is included in a complete interaction trace.

- ∀ $m_i \in$ output(S_i), if polarity (m_i,P) = - then ∃ $I(S_i, S_{i+1}, m_i^+, tc) \in T$ and ∃ $I(S_i, S_{i+1}, m_i^-, tc) \in T$ where $C_i^- \subseteq pl_i^+$, and ∃ $I_p(S_i, S_i, \overline{m_i}, t_{i+1}, S_{i+1}, S_{i+1}) \in T_p$, and $(S_{i+1},S_{i+1}) \in R$, which implies that there is a new transition

added to the partial interaction trace *tr* is (S_i, S_{i+1}, $\overline{m_i}$,t_i,S_{i+1},S_{i+1}) and because (S_{i+1},S_{i+1}) is co-accessible(there is a path to the final state) then the interaction trace *IT*(*tr*.(S_i, S_{i+1}, $\overline{m_i}$,t_i,S_{i+1},S_{i+1}) can reach a final state. Therefore, *IT*(*tr*.(S_i, S_{i+1}, $\overline{m_i}$,t_i,S_{i+1},S_{i+1}) is start by initial state and ends by final state so it is included in a complete interaction trace.

Proof 6-2: Compatibility in terms of interaction trace implies compatibility in terms of automata.

Given two TBP $P(S,s_0,T,X,F)$ and $P'(S',s'_0,T',X',F')$ are compatibles in terms of interaction trace *IT*.

- If ∀ *IT* that starts at (S_0,S'_0) and ends (S_{n+1},S'_{n+1}), (i.e ∀ $tr \in Tr_p(P,P')$

with tr =(S_0,S'_0,m_1, t_1, S_1, S'_1), □ ., (S_{n-1},S'_{n-1},m_n,t_n, S_n ,S'_n))

  - If ∀ m_{n+1} and ∀t_{n+1} if ∃ (S_n,S_{n+1},m^-_{n+1},$tc_{(n+1)}$) ∈ *T* then ∃ $s'_{n+1} \in S'$

where the interaction trace *IT*(*tr*.(S_n, S'_n, $\overline{m_{j+1}}$,t_{n+1},S_{n+1},S'_{n+1}) is included in a complete interaction trace. Transition(S_n, S'_n, $\overline{m_{j+1}}$,t_{n+1},S_{n+1},S'_{n+1}) in the previous interaction trace implies that in all partial interaction trace transition which resulted from the transition (S_n,S_{n+1},m^-_{n+1},$tc_{(n+1)}$) ∈ *T* where $C_i^- \subseteq pl_i^+$. So, this presents the first property of a relation *R* where (S_n,S'_n) and (S_{n+1},S'_{n+1}) ∈*R*.

Since $IT(tr.(S_n, S_{n}, \overline{m_{j+1}}^{+}, t_{n+1}, S_{n+1}, S_{n+1})$ is includes in a complete interaction trace so (S_n, S_{n}) and (S_{n+1}, S_{n+1}) are co-accessible which is a second property of the relation *R*. This interaction trace is starts by the initial state, so there is a path from it to the final state and because *tr* is included in complete interaction trace so this state is co-accessible which is a third property of the relation *R*.

- If $\forall m_{n+1}$ and $\forall t_{n+1}$ if $\exists (S_{n}, S_{n+1}, m_{n+1}, tc_{(n+1)}) \in T$ then $\exists s_{n+1} \in$

S, where the interaction trace $IT(tr.(S_n, S_{n}, \overline{m_{j+1}}, t_{n+1}, S_{n+1}, S_{n+1})$ is included in a complete interaction trace. Transition $(S_n, S_{n}, \overline{m_{j+1}}, t_{n+1}, S_{n+1}, S_{n+1})$ in the previous interaction trace implies that in all partial interaction trace transition which resulted from transition

$(S_{n}, S_{n+1}, m_{n+1}, tc_{(n+1)}) \in T$ where $C_i \subseteq pl_i^+$. So this presents the

first property of a relation *R* where (S_n, S_{n}) and $(S_{n+1}, S_{n+1}) \in R$.

Since $IT(tr.(S_n, S_{n}, \overline{m_{j+1}}, t_{n+1}, S_{n+1}, S_{n+1})$ is included in a complete interaction trace, so (S_n, S_{n}) and (S_{n+1}, S_{n+1}) are co-accessible which is a second property the relation R. This interaction trace is starts by the initial state, so there is a path from it to the final state and because *tr* is included in complete interaction trace so this state is co-accessible which is the third property of the relation *R*.

As a result, each state in the partial interaction trace is based on the previous definition of the compatibility in terms of interaction trace is belongs to the relation R.

And all the properties required in *R* are satisfied by the definition of the compatibility in terms of interaction trace.

6.3. Implementation

The compatibility and replaceability algorithms are implemented and used as a verification tools on the COMPAS[7] (Compliance-driven Models, Languages, and Architectures for Services) project. This software package consists of an eclipse plug-in for editing business protocols and a library for checking compatibility and replaceability between web services using their business protocols. The access control specification, as an example of data constraints, is presented using the ontology in this package.

In our implementation, the ontology is created in a (*.owl). For example, Figure 6-9 describe an ontology taxonomy and Listing 1 presenting the (*.owl) file contents of this ontology. As it is possible to see in the last part of the Listing 1, there are some individuals that belong to the class taxonomy.

```
<?xml version="1.0"?>
<rdf:RDF
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:protege="http://protege.stanford.edu/plugins/owl/protege#"
xmlns:xsp="http://www.owl-ontologies.com/2005/08/07/xsp.owl#"
xmlns:owl="http://www.w3.org/2002/07/owl#"
xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
xmlns:swrl="http://www.w3.org/2003/11/swrl#"
xmlns:swrlb="http://www.w3.org/2003/11/swrlb#"
xmlns="http://www.owl-ontologies.com/prova.owl#"
xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
xml:base="http://www.owl-ontologies.com/prova.owl">
<owl:Ontology rdf:about=""/>
<owl:Class rdf:ID="Journal_papers">
<rdfs:subClassOf>
<owl:Class rdf:ID="Library_papers"/>
</rdfs:subClassOf>
```

.http://www.compas-ict.eu[7]

```
</owl:Class>
<owl:Class rdf:ID="card"/>
<owl:Class rdf:ID="Professor_card">
<rdfs:subClassOf rdf:resource="#card"/>
</owl:Class>
<owl:Class rdf:ID="University_student_card">
<rdfs:subClassOf>
<owl:Class rdf:ID="Student_card"/>
</rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Conference_paper">
<rdfs:subClassOf rdf:resource="#Library_papers"/>
</owl:Class>
<owl:Class rdf:about="#Student_card">
<rdfs:subClassOf rdf:resource="#card"/>
</owl:Class>
<owl:Class rdf:ID="School_card">
<rdfs:subClassOf rdf:resource="#Student_card"/>
</owl:Class>
<owl:SymmetricProperty rdf:ID="hasCredential1">
<rdfs:range rdf:resource="#card"/>
<rdfs:domain rdf:resource="#Journal_papers"/>
<rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
<owl:inverseOf rdf:resource="#hasCredential1"/>
</owl:SymmetricProperty>
<owl:SymmetricProperty rdf:ID="hasCredential">
<rdfs:domain rdf:resource="#Conference_paper"/>
<rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
<owl:inverseOf rdf:resource="#hasCredential"/>
<rdfs:range>
<owl:Class>
<owl:unionOf rdf:parseType="Collection">
<owl:Class rdf:about="#Professor_card"/>
<owl:Class rdf:about="#Student_card"/>
</owl:unionOf>
</owl:Class>
</rdfs:range>
</owl:SymmetricProperty>
<Library_papers rdf:ID="L1"/>
```

```
<Student_card rdf:ID="S1"/>

<Journal_papers rdf:ID="J1"/>

<Conference_paper rdf:ID="C1"/>

<Professor_card rdf:ID="P1"/>

<card rdf:ID="c1"/>

<School_card rdf:ID="Sc1"/>

<University_student_card rdf:ID="U1"/>

</rdf:RDF>
```

Listing 1: Representation of the (*.owl) file of the AC ontology presented in Figure 6-9

Chapter 7 . GENERAL SPECIFICATION APPROACH

In the previous chapters, we discussed the modeling and analyzing of Web services interoperability in the presence of some important parameters such as time and AC. This chapter presents a general approach in which we can perform our analysis in the presence of many parameters.

7.1. Introduction

The interaction between the Web services depends on the behavior of each service. This behavior is presented by the business protocol. If the Web service is considered as a Web resource, then the business protocol of it can be seen as a general constraint that must be satisfied during the interaction. The business protocol can be simple and only includes the order of the messages that can be presented by the service. But the majority of service includes more constraints such as the access control policies and the time constraints. Therefore, there is a crucial need to enrich the business protocol with all the possible constraints that are required during the interaction. As a result, modeling and analyzing web service behaviors after including all the possible constraints needs general approach. In this approach, we will model and analyze the Web service behavior after including all types of the constraints. Thus, each transition in the business protocol is assigned by a message with a set of constraints. We will call this "message specification" The message specification consists of a message name and a set of attributes such as XMLSchema, ACP, credentials, privacy, meaning, and time.

7.2. Abstract interpretation

Model Checking technique takes into account every possible state of the system and determines if it is consistent with the designer's specifications. But, this technique is limited by the size of the systems it can analyze. For example, the concrete semantics model of a program (the set of all its possible executions in all execution

environments) is not computable. All the non trivial questions on the concrete program semantic are undicidable.

Abstract Interpretation [52], by contrast, doesn't attempt to look at every possible state of a system, but to develop a simplified approximation of a system that preserves the particular properties that need to be assessed. This makes it possible to analyze very large, complex systems, but with less precision than is possible with model checking.

Each message specification constraint is an abstract domain for a set of instances values (concrete values) and the relation between these constrains and its instance values is a Galois connection relation [51] which is a part of the abstract interpretation theory.

***Definition 7-1**. **Galois connection:** (A, ≤) and (B,⊑) are posets; a pair α :A→B*

and γ :B→A is a Galois connection iff {∀u ∈ A, ∀v ∈ B, α(u)⊑v ≡ γ(v)} and is

written as $(A,\leq)\underset{\gamma}{\overset{\alpha}{\leftrightarrows}}(B,\leq,\sqsubseteq)$.

In the Galois connection, the functions α and γ are called concretization and abstraction functions respectively. In our approach, each specification value is presented as an abstract domain and the concrete domain is the set of the instances that satisfy this specification.

7.3. Formalizations

Each message specification consists of a set of attributes presenting the different types of the constraints. These specifications attributes have domains of specifications. There is a set of operations that can be applied on these attributes such as subsumption, intersection, and composition.

***Definition 7-2: Specification attribute domain** E is consists of:*

- *A lattice $(SV, \sqcup_{SV}, \sqcap_{SV}, \subseteq_{SV})$ of specification values (SV)*

- *A set I of instance values*
- *An instantiation morphism, $f_i : SV \rightarrow 2^I$*

- *Subsumption partial order $\subseteq$ on SV, such that $f(sv) \sqsubseteq f(sv') \Leftrightarrow SV \subseteq SV'$*

- *An annotation function : $f_a : XML^* \rightarrow (XML \times I)^*$, where XML refers to the set of all message in an xml format.*
- *Annotates the last element of the sequence: if $f_a(xms.xm)=XMS.XM$, then $f_a(xms)=XMS$, where xms refers to the sequence of messages before annotation, xm refers to an xml message before annotation, XMS refers to the sequence of messages after annotation, and XM refers to an xml message after annotation.*

The relation between the lattice $(SV, \sqcup_{SV}, \sqcap_{SV}, \subseteq_{SV})$ of specification values and the lattice $(2^I, \sqsubseteq)$ of instances value is a Galois connection. If the specification doesn't contain some attributes, we will put default values for these attribute. The credentials specification attribute domain *E(Credentials) ={Visa, Master Card, ID Card, Driver license,...}* is an example of a specification attribute domain. The set of the specification values of the credentials is a poset and each specification value has a set of instances. The subsumption between the specification values means that the set of the instances of the subsumed specification value is included in the

set of the instances of the other specification value. The annotation process deals with the instance values and in each transition the current instance value is annotated to the message with the previous values that are provided in the previous transitions. For example, the credentials instance values on each transition are the cumulated credentials which are provided in the previous transitions and instance value in the current transition.

Each transition of the business protocol will have different specification attributes such as the ACP and time. The specification attribute domain of these specification attributes is the composition of them.

Definition 7-3: Composed specification attribute domain *is a tuple* (SV_{cas}, I_{cas}, F_{icas}, F_{acas}) *where:*

- $Poset(SV_{cas}, \subseteq_{Scas})=(SV_{as1} \times SV_{as2} \times \square .. \times SV_{asn}, \subseteq_{Scas})$

  Where $(SV_{as1},..., SV_{asn}) \subseteq_{Scas} (SV\square_{as1},..., SV\square_{asn})$

  Iff $SV_{as1} \subseteq_{as1} SV\square_{as1}$ *and* $\square$ *and* $SV_{asn} \subseteq_{asn} SV\square_{asn}$

- $I_{cas}=I_{as1} \times \square \times I_{asn}$, $xm.y \rightarrow XM.(y,L_y)$
- $F_{icas}(SV_{as1}\square\ SV_{asn})=f_{ias1}(SV_{as1})\times\square..\times f_{iasn}(SV_{asn})$, $xm \rightarrow XM$
- $F_{acas}(xm)=(f_{aas1}(xm), \square, f_{aasn}(xm), \ xm.y \rightarrow L_y$

An example of a composed specification attribute domain is E_{cas}={*ACP=(Visa, ID Card, Master Card, Driver license,* $\square$*), Time=(t1*$\in$ *[1,*∞*[, t2*$\in$ *[2,10[,* $\square$ *), Mean-*

ing=(request ID , Response, confirmation ,□.), XMLSchema=(m1.xsd, m2.xsd,□), (Credentials=(Student Card, Visa,□)}. This attribute domain E_{cas} contains a set of specification attribute domains for a set of specification values such as the ACP, Time, Meaning, XMLSchema, and Credentials. Each specification value has its specification attribute domains. For example, the specification attribute domain of the ACP specification value is (*Visa, ID Card, Master Card, Driver license, □)* and the specification attribute domain of the XMLSchema specification value is *(m1.xsd, m2.xsd,□)*. The initiation function assign for each composed specification value a set of instances by assigning to each specification value the set of instance that satisfy this specification.

Definition 7-4: (Message specification (ms) *and* ***Message instance (m))*** *ms is a set of specification values associated to the message and m is a concrete message that matches certain message specification*

An example of message specification is *ms={ACP=(Visa), Time=(t1∈ [1,10[ ),*

Meaning = (confirmation), XMLSchema=(m1.xsd), (Credentials=(False)}. The concrete message *m* is the message that satisfies all the previous constraints by containing instance values satisfy all the specification values. In other words, the message *m* must contains a visa card , the value of the clock *t1* must be between the *one* and the *ten* when this message is sent, the meaning is a confirmation, and the structure of the message *m* must be compatible with the schema described in the *m1.xsd* file.

Definition 7-5: Message annotation *is the process of annotating sequence of massages matches a service protocol using annotation function* f_a*: (XML* $\times$ *T)* $^{*}\rightarrow I$.

Each attribute (constraint type) has a way of annotating; ACP attribute value of a message is annotated as the union of all ACP declared in the previous messages and did not receive its equivalent credentials from the other party. The credential can be annotated as the sum of all the provide credentials in the previous message.

Definition 7-6: Annotated message as an instance of message specification, *We say that an annotated message m1 with attributes* (A_1 (v_1), A_2(v_2), □ A_n(v_n)) *, where* (A_1 , A_2, ..A_n) *are attributes and* v_1,v_2,v_3 *are annotated values for these attributes,*

is an instance of message specification $(n,SV_1(c_1),\ldots, SV_n(c_n))$*, where n is the specification name and* $c_1,c_2,..c_n$ *are the sets of instances values for each attribute ,*

if $c_j=F_{Ej}(SV_j)$ *and* $v_j \in c_j$.

For example, the set of instances for the *XMLSchema= (m1.xsd)* includes all the instance XML messages that can be validated by the XMLSchema which is present in the *m1.xsd* file and the set of instances for the *ACP* =(*Visa*) is all the instance that have a visa card and the composed instance set is all the instances that satisfy the specification.

The new definition of the business protocol includes the constraints in terms of message specification. We call this type of protocols General Constraints Business Protocols From the point of view of message specification definition, time is considered as one of the attribute specification domain. Therefore; the definition of the constrained business protocol does not explicitly mention the time but embeds it in the message specification.

Definition 7-7: General Constraints Business Protocol (GCBP) *is a tuple P= (S,* s_0*, T, X, F) where:*

- *S is a finite set of states.*
- s_0 *is the start state of protocol P and* $S_0 \in S$.
- *X is a finite set of clocks with set of clock reset* $CR \in X$.

- *T is a finite set of transitions with :* $T \subseteq S^2 \times MS \times \{-,+\} \times CR \times RK$, *a finite set of explicit transition where MS is the set of message specification (ms) assigned to the explicit transitions and CR is the set of clock reset. RK is the set of variables reset, if there is no dynamic variables in the protocol, then this set is always empty.*

- $F \subseteq S$ *is the set of final states.*

Definition 7-8: Attribute subsumption, *attribute E with set of instances* I_1 *is subsumed by attribute E with set of instances* I_2 *if* I_1 *is subset of* I_2. *If the specification doesn't contain some attributes, we will put default values for these attributes.*

The default values of the attribute depend on its type. For example, if there is no ACP, the default value is "true" and for the credentials the default value is "False". In the case of time, the default value is the time interval from zero to infinity.

Figure 7-1 shows an example of two Web services business protocol with general constraints messages specifications. Each business protocol is consists of a set of states with a start state and an end state and a set of transitions with a set of clocks reset and message specifications. The message specifications on each transition consist of a set of attributes such as the ACP, credentials, time, XMLSchema, and meaning. Each attribute is assigned with a specification value or a default value. For instance, the transition between the states *s2* and *s3* in the protocol *P1* has a message specification value :(***ResourceReqSent(-) (XMLSchema {m3.xsd}, Credentials{False,}, Meaning {resource request }, ACP{True}, time{t1[0,10[})*** with a default values for the ACP={*True*} and Credentials={*False*} which means that the service in this transition does not require an access control policy and does not provide credentials respectively.

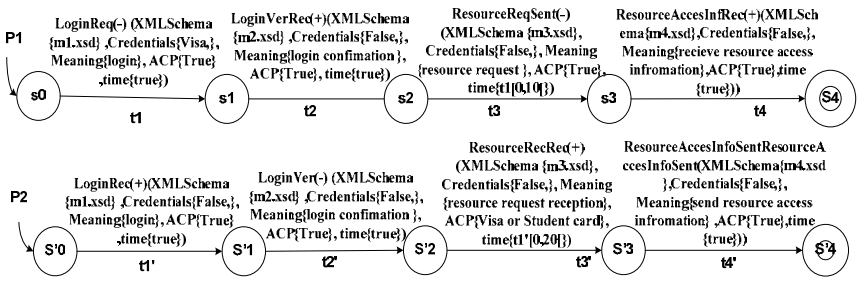

Figure 7-1. Two Web services business protocols with general constraints messages specifications**.**

Definition 7-9: (Messages Specification subsumption) *message specification ms1 with set of attribute (E_1, E_2, E_n) is subsumed by message specification ms2 with set of attributes (E'_1, E'_2, E'_n) if for each $1 \leq i \leq n$, and $1 \leq j \leq n$, $\exists E_i \subseteq E'_j$.*

An example of message specification subsumption can be seen between the message specification (*ms1*) on transition between the states *s0* and *s1* in the protocol *P1* and the message specification (*ms1'*) on the transition between the states *s'0* and *s'1* in the protocol *P2* in Figure 7-1 where *ms1* $\subseteq$ *ms1'*. We can notice that the credentials attribute in the message specification *ms* is (**Visa)** but the ACP has the default value (**False)** which means that the first service provides a credential and the other service does not need credentials in this transition.

Definition 7-10: Messages specification intersection (ms ∩ ms') *is a message specification ms'' with a composed attribute specification such that each attribute value in this specification is the lower attribute value of each pairs in the argument messages ms and ms' by comparing the tuple of theses messages as to point comparison.*

The message specifications intersection is used in the process of the product automata creation of two GCBP. This intersection produces a new message specification with the set of instances that is included in the original specifications.

Definition 7-11: *Conversation is a sequence of XML messages instances.*

Definition 7-12: (Conversation w.r.t business protocol) *a correct conversation w.r.t BP is a sequence of XML messages instances that can be recognized by the protocol where each message instance satisfies the corresponding message specification.*

Definition 7-13: Interaction trace between two protocols P and P′: *Interaction trace IT is a sequence of the form* $((S_i, S'_i, \overline{m}_{i+1}, S_{i+1}, S'_{i+1}))_i$ *where* $S_i, S_{i+1} \in S$ *of protocol P, and* $S'_i, S'_{i+1} \in S$ *of protocol P′ and ,* $\overline{m}_{i+1}$ *define the annotated message on the transition and the two direction arrow means that it can be input or output depending on the polarity of output messages from states* S_i *and* S'_i.

Definition 7-14: Correct interaction trace w.r.t two protocols*: Interaction trace IT is a sequence* $((S_i, S'_i, \overline{m}_{i+1}, S_{i+1}, S'_{i+1}))_i$. *An IT matches a conversation between two GCBP* $P(S, S_0, T, F)$ *and* $P'(S', S'_0, T', F')$ *if* $\forall (S_j, S'_j, \overline{m}_{j+1}, S_{j+1}, S'_{j+1})$ *in IT where* $\overline{m}_{j+1} \in (ms_{j+1} \cap ms'_{j+1})$.

- *If* $\overline{m}_{j+1} = \overline{m_{j+1}}$ *then*

- $\exists (S_j, S_{j+1}, ms_{j+1}^-) \in T$.

- $\exists (S_j, S_{j+1}, ms_{j+1}^+) \in T$

- *If* $\overline{m}_{j+1} = \overline{m}_{j+1}$ *then*

- $\exists (S_j, S_{j+1}, ms_{j+1}^+) \in T$.

- $\exists (S_j, S_{j+1}, ms_{j+1}^-) \in T$

Definition 7-15:(Compatibility in Terms of Interaction Trace) *Two constrained business protocol, P(S, S_0, T, X, F) and P'(S', S'_0, T', X', F') are compatibles in terms of interaction trace IT*

If $\forall$ *IT that starts by* S_0, S'_0 *and ends* S_n, S'_n *(i.e* $\forall$ $tr \in Tr_p(P, P')$ *with tr*

$= (S_0, S'_0, \overline{m}_1, S_1, S'_1), \ldots, (S_{n-1}, S'_{n-1}, \overline{m}_n, S_n, S'_n)$

- $\forall\ m_{n+1} \in ms^{-}_{n+1}$, *if* $\exists(S_n,S_{n+1},ms^{-}_{n+1}) \in T$ *then* $\exists\ s_{n+1} \in S$ *and* $\exists(S_n,S_{n+1},ms^{+}_{n+1}) \in T$ *such that* $IT(S_n, S_n, \overline{m_{n+1}},S_{n+1},S_{n+1}) \in Tr_p(P,P)$ *and* $\overline{m_1} = \overline{m_{n+1}}$ *and the interaction trace* $IT(tr.(S_n, S_n, \overline{m_{n+1}},S_{n+1},S_{n+1}))$ *is included in a complete interaction trace*

- $\forall\ m_{n+1} \in ms^{+}_{n+1}$, *if* $\exists\ (S_n,S_{n+1},ms^{-}_{n+1}) \in T$ *then* $\exists s_{n+1} \in S$ *and* $\exists\ (S_n,S_{n+1},ms^{+}_{n+1}) \in T$ *such that* $IT(S_n, S_n, \overline{m_{n+1}},S_{n+1},S_{n+1}) \in Tr_p(P,P)$ *and* $\overline{m_1} = \overline{m_{n+1}}$ *and the interaction trace* $IT(tr.(S_n, S_n, \overline{m_{n+1}},S_{n+1},S_{n+1}))$ *is included in a complete interaction trace.*

The product automata definition supposes that the pre-processing operations such as the removal operation of the implicit transitions of the business protocols are performed.

Definition 7-16. (Product automata of two GCBP) *$P(S, S_0, T, X, F)$ and $P'(S',S'_0, T', X', F')$ are two CBP, we construct product automata of P and P', $A^p=P \times P' =(S^p,S_o^p,T^p, F^p)$ where:*

- $S^p \subseteq S \times S'$

- $S_o^p \subseteq (S_0, S_0)$

- $T^p \subseteq ((S \times S) \times (S \times S) \times (MS \cap MS))$ such that for all transition $((S_i, S_i), (S_{i+1}, S_{i+1}), \overline{ms}_{i+1}) \in T^p$ there exist two transitions $(S_i, S_{i+1}, ms_{i+1}) \in T$ and $(S_i, S_{i+1}, ms_{i+1}) \in T$ with:

  - $\overline{ms}_{i+1} = (ms_{i+1} \cap ms_{i+1})$ is the message specification which resulted from the intersection of the ms_{i+1} and ms_{i+1} message specifications.
  - $Polarity(ms_{i+1}, P) \neq Polarity(ms_{i+1}, P)$
    - If $polarity(ms_{i+1}, P) = -$ then $\overline{ms}_{i+1} = \overline{ms_{i+1}}$, and $polarity(ms_{i+1}, P) = +$.
    - If $polarity(ms_{i+1}, P) = -$ then $\overline{ms}_{i+1} = \overline{ms_{i+1}}$, and $polarity(ms_{i+1}, P) = +$.

- $F^p \subseteq F \times F$

Figure 7-2 shows the product automata of the two business protocols *P1* and *P2* of Figure 7-1. In this automata, the name of each state is the names of the two states of the corresponding states of the two protocols (e.g., the start state (*s0*,*s'0*)). Each transition contains a message specification name and the message specification which is the result of the intersection between the two corresponding message specifications of the two protocols. The direction of the arrow on the message indicates the sender and the receiver of the message from the two protocols in this transition. If the arrow is from left to right this means that the message will be sent from the protocol which has the first state in the state name. For example in the transition between the states (*s0*,*s'0*) and (*s1*,*s'1*) the arrow is from the left to right which means that this message is sent from the protocol *P1* to the protocol *P2* and vice-versa. The produced product automata in Figure 7-2 shows that the time intersection in the transition between the states (*s2*,*s'2*) and (*s3*,*s'3*) is ((*t1*,*t1'*)[0,10[). The time intersection is calculated directly because the two clock reset *t1* in *P1* and *t'1* in *P2* are corresponding to each other.

Before performing the compatibility checking, some attributes needs to be cumulated. The most common example for these attributes is the credentials attribute. The cumulating process enable the checker to identify in each transition what exactly the provided information (credentials). For instance, in the protocol *P1* in Figure 7-1 the **Visa** credential is provided in the first transition and as a result this credential is included implicitly in the credentials attributes in the next transitions. Therefore, the credential attributes on the next transition that are used in the interaction must be updated by adding the previously provided credentials. This step is performed after the creation of the product automata. The cumulating process for credentials is explained in details in [63].

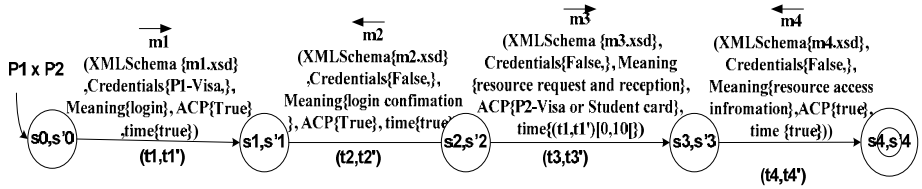

Figure 7-2. The product automata of the two business protocols P1 and P2 of Figure 7-1.

The algorithm for cumulating one attribute is different than the algorithm for cumulating more than one attribute. In case of cumulating more than one attribute

on the message specification, the cumulating process cannot be applied on each attribute separately. These attributes must be cumulated as combined attribute (i.e., the attributes that need cumulating in the message specifications will be put in a tuple and treated together as one attribute during the cumulating process). Figure 7-3 shows the product automata of Figure *7-2* after cumulating the credentials.

***Definition 7-17. (Cumulative path in the product automata with more than attribute)**: $PA^p = ((s_i^1,s_i^2)$ $\xrightarrow{(C_{00}^1,\dots,C_{l0}^1)(C_{00}^2,\dots,C_{l0}^2)}$ (s_{i+1}^1,s_{i+1}^2) ; □ ;(s_n^1,s_n^2) $\xrightarrow{(C_{0m}^1,\dots,C_{lm}^1)(C_{0m}^2,\dots,C_{lm}^2)}$ $(s_{n+1}^1,s_{n+1}^2))$ is a cumulative path in the product automata $A^p = (S^p; s_0^p; T^p; F^p)$ where*

□*l is the number of the attributes that need cumulating on the message specification.*

□*States $(s_i^1 ; s_i^2)$;* □ *; $(s_n^1; s_n^2) \in S^p$.*

□*Each attributes $(C_{0t}^1,\dots,C_{lt}^1)$ is the set of cumulative attributes which is the union of the previous set of cumulative attributes $(C_{0(t-1)}^1,\dots,C_{l(t-1)}^1)$ and the current set of attributes $(c_{0t}^1,\dots,c_{lt}^1)$ where $(c_{0t}^1,\dots,c_{lt}^1)$ is the set of attributes on the transition between the state (s_i^1 and s_{i+1}^1) of the protocol P^1 and $(C_{00}^1,\dots,C_{l0}^1) = (c_{00}^1,\dots,c_{l0}^1)$ and each attribute $(C_{0t}^2,\dots,C_{lt}^2)$ is the set of cumulative attributes which is the union of the previous set of cumulative attributes $(C_{0(t-1)}^2,\dots,C_{l(t-1)}^2)$ and the current set of attributes $(c_{0t}^2,\dots,c_{lt}^2)$ where $(c_{0t}^2,\dots,c_{lt}^2)$ is the set of attributes on the transition between the state (s_i^2 and s_{i+1}^2) of the protocol P^2 and $(C_{00}^2,\dots,C_{l0}^2) = (c_{00}^2,\dots,c_{l0}^2)$.*

□*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 $(s_f^1 ; s_f^2) \in F^p$.*

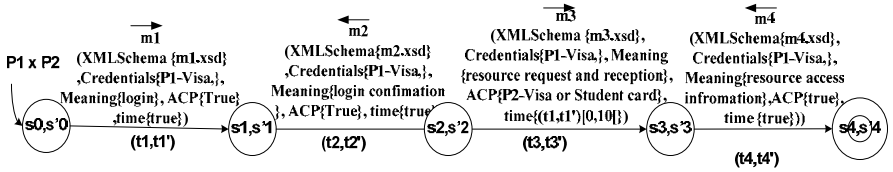

Figure 7-3.The product automata of the two business protocols *P1* and *P2* of Figure 7-1 after cumulating the credentials.

Definition 7-18*:(path in the Product of Two GCBP),* *The $A^p(S^p,S_o^p,T^p, F^p)$ is the automat product of two CBP, the path $C=S_1,S_2,\square.,S_n$ is a succession of states of S^p , where $\forall S_i$ and $S_{i+1} \in C$, $\exists (S_i, S_{i+1,} m_{i+1}) \in T^p$. Path is complete if it start by initial state and ends by final state. The concatenation of two paths $C_1= S_1,S_2,\square.,S_i$ and $C_2= S_j,S_{j+1},\square.,S_n$ if $\exists I(S_i, S_{j,} m_i) \in T^p$. This is can be written as $C_1.C_2$. If C and C□ are two paths in S^p where |C□| <=|C|, we say that C□is included path of C and can be noted as C□⊑ C , if there exist two paths C_1 and C_2 of S^p such that C=C_1.C□.C_2*

Definition 7-19. *Each partial interaction trace is a path in the product automata.*

Proof:

Put protocols P (S, S_0, T, X, F) and P□(S□,S□$_0$, T□, X□, F□) are two CBP and tr =(S_0,S□$_0$,m_{t1},S_1, S□$_1$), (S_1,S□$_1$,mt_2,, S_2, S□$_2$) ,□, (S_{i-1},S□$_{i-1}$,m_{ti}, S_i,S□$_i$) is a partial interaction trace between two protocol, this partial interaction trace present the inter-

action between two sequence of states and their messages(input/output) ,conversation, between two protocols. For instance, In the first protocol this sequence is ($S_0 \rightarrow S_1 \rightarrow \ldots S_i$. and in the second is $S'_0 \rightarrow S'_1, \rightarrow \ldots S'_i$ if we determinate the product automata of these two sequences, the result will be (S_0,S'_0, M_{p1},S_1, S'_1), (S_1,S'_1,M^p_2, S_2, S'_2) , … , (S_{i-1},S'_{i-1},M^p_i, S_i,S'_i) , since from the definition of interaction

trace $mt_1 \in M_1 \cap M'_1, \ldots$, $m_{ti} \in M_i \cap M'_i$ and from the definition of product auto-

mata, the message specification $M^p_1 = M_1 \cap M'_1, \ldots$, $M^p_i = M_i \cap M'_i$ which leads to

$m_t \in M^p$ and that each partial interaction trace is a path in the product automata.

Definition 7-20. Co-accessibility of a State in the Product of Automata

$A^p=(S^p,S_o^p,T^p,,F^p)$ is the product of automata of two CBP P and P', state $s_i^p \in S^p$ is

co-accessible if there exist two paths c and c' where $c = s_i^p.c.s^f$ and $s_f \in F^p$.

The compatibility in terms of product automata definition supposes that all the pre-processing operations are performed. These pre-processing operations include the calculation of the cumulative ACP and the removal of the implicit transitions. Therefore; the comparison operations between message specifications in the compatibility definition suppose that these message specifications are updated by all the necessary pre-processing operations.

Definition 7-21. (Compatibility in terms of Product Automata) *Protocols P (S, S_0, T,X, F) and P'(S',S'_0,T', X', F') are two GCBP, and $A^p=(S^p,S_o^p,T^p, F^p)$ is their*

product automata, we say that P and P′ are compatible using their product automata if there is a relation $R \subseteq S \times S'$ *where for all* $(S_i, S'_i) \in R$:

- $\forall ms_i \in output(S_i)$, *if polarity* $(ms_i, P) = -$ *then* $\exists (S_i, S_{i+1}, ms_i^-) \in T$ *and* $\exists (S'_i, S'_{i+1}, ms_i^+) \in T'$ *where* $ms_i^- \subseteq ms_i^+$ *and* $\exists (S_i, S'_i, \overline{ms_i}, S_{i+1}, S'_{i+1}) \in T^p$, *and* $(S_{i+1}, S'_{i+1}) \in R$.

- $\forall ms'_i \in output(S'_i)$, *if polarity* $(ms'_i, P') = -$ *then* $\exists (S'_i, S'_{i+1}, ms'^-_i) \in T'$ *and* $\exists (S_i, S_{i+1}, ms_i^+) \in T$ *where* $ms'^-_i \subseteq ms_i^+$ *and* $\exists (S_i, S'_i, \overline{ms_i}, S_{i+1}, S'_{i+1}) \in T^p$, *and* $(S_{i+1}, S'_{i+1}) \in R$.

- *$(S_i,S'_i) \in S^p$ is co-accessible (there is a path in the product automat from this state to final state.*

- *$(S_0, S'_0) \in R$.*

7.4. Adaptive compatibility

The type of the constraint in each message on the Web services determines the way of the annotations of the message for this type. Thus, we categorized the types of constraints to more than one type. The first type is the constraints that are depending only on the instances values that will be provided in the current message. We will call this type "history independent constraints". The second type of constraints depend on the instances values that are provided before this message. We will call it "history dependent constraints". In this type, the instances values can be cumulative. For instance the ACP constraints on a message depend on the cumulative credentials of the previous messages. Time constraints are another type of constraints which need a special treatment. Due to these different types of manipulations, checking the compatibility with all of these different types of constraints in the same time in terms of complexity is exponential and in some cases undecidable. Therefore, we propose the fine-grained compatibility checking approach for message specifications. In this approach, we decide before checking the compatibility the types of the constraints which are needed to check and choose the algorithm based on this. The choice of the compatibility checking algorithm depends on the types of the constraints in which we want to check. The history independent constraints algorithm is applied in the first step. If the two services are compatible then the other algorithms are applied.

The history independent constraints compatibility checking algorithms check the message specification attribute constraints that do not depend on the history of the provided instances. The message schema attribute constraint is an example of these

type constraints. The instance messages of these schemas depend only on the current message. This algorithm is a simple form of the algorithm which is used for checking the compatibility of Web service with AC after removing the AC part. This means that there is no need for calculating the cumulative ACC.

In the history dependent constraints, the compatibility checking algorithms check the message specifications after updating the set of possible messages instances. The updating can be cumulative as in the case of the ACC. For all cumulative attribute constraints we can use the previous algorithms for checking the compatibility and the replaceability with AC.

Time constraints in the message specification are a special case for compatibility and replaceability because time analysis varies from simple computation complexity to a very complex computation complexity. The algorithms for the time analysis are discussed in details in the previous chapter.

Chapter 8 .WEB SERVICES CHOREOGRAPHY

This chapter discusses the Web service selection for choreography implementation based on the compatibility checking with access control. It starts by an overview about the Web services choreography with AC. Then, it lists the formalizations which includes the modeling of the Web services choreography and the compatibility checking analysis. Finally, the verification process is explained and the related works are presented.

8.1. Web service choreography with AC

The standardization in the Web services makes them reusable in different ways through the Web. For example, the Web service can be used alone or composed with other services for performing a specific operation. Web services choreography is used in the design phase of complex peer-to-peer applications in which each peer can be implemented by a Web service. The behavior of each peer (service) must be specified in the choreography of the application. The Web services that are needed to join the choreography must conform to that specification. Several research efforts focus on the issue of determining whether the behavior of the Web services implementing choreography matches the one described by the choreography specification (Paci et al. [107]; Busi et al. [43] ;Kazhamiakin and Pistore[89].

Behavior conformance must include the satisfaction of the access control policies (ACP) between Web services. In this chapter, Web service behavior description is enriched by annotating the AC on the business protocols. Moreover, we assume that the invocation of each Web service operation is controlled by access control policies; such policies establish which credentials the invoker Web service must possess in order to be able to invoke the operation.

A business protocol of a service is presented in the previous chapters but in this chapter its definition is extended by specifying on each transition the name of the service that will receive this message (if this message is an outgoing) or send this message (if this message is incoming).

This chapter discusses the results of modeling and analyzing Web services business protocols augmented with access control policies to verify the implementation of Web services choreography. Access control policies are expressed using ontology 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) *n* services can have interactions based on their business protocols.

Ontology is an explicit specification of a conceptualization [77]. It defines a common vocabulary for researchers who need to share information in a domain. It includes machine-interpretable definitions of basic concepts in the domain and relations among them. We will use the ontology in presenting the ACP and credentials (as shown in the previous chapters).

8.2. Formalization and Algorithms

8.2.1 Modeling Web services choreography

Web service choreography is modeled as a non deterministic transition system. In this model, business protocol defines Web service behavior that must be followed by the selected web service to participate on the choreography. Each state of the model presents the status of each business protocol. The transitions between the states of the model are annotated by the messages exchange between services with ACP or credentials. Each message exchange is associated with an operation offered by a service and implies an exchange of information between the invoker service and the service providing the operation. The message's sender and receiver are also annotated on the transition. A c*onversation* is a sequence of message exchanges starts by the start state and ends by a final state.

Definition 8-1: (WS-Choreography Transition System with AC) *A choreography is represented by a non deterministic transition system TS = (S, M, T, s_o, s_f). S is a set of choreography states. Each state is a tuple of the form (Na, (p_1, $state_1$),(p_2, $state_2$),...,(p_n, $state_n$)) where Na is the state name and in each tuple (p_i, $state_i$), $state_i$*

represents the state of business protocol p_i. $s_0 \in S$ is the initial state and $s_f \in S$ is the final state. M is a set of message exchanges. Each message exchange is represented by a tuple (p_s, p_d, o, AC, c, mt), where p_s is the business protocol of the message's sender, p_d is the business protocol of the receiver of the message, o is the operation of the p_d that invoked by p_s that triggers the message exchange, AC is the access control policy, c is the set of credentials, and mt is the message type. $T \subseteq S \times M \times S$ is the transition relation. A transition (s, m, s') $\in T$ if $m = (p_s, p_d, o, AC, c, mt)$ and the tuples (p_s, $state_s$) and (p_d, $state_d$) in state s are replaced by the tuples (p_s, $state'_s$) and (p_d, $state'_d$) in state s' (respectively) due to the invocation of the operation o.

Example 8-1: Figure 8-1 shows an example of Web service choreography describes shopping process between *seller*, *buyer*, and *broker*. *Credit agency* partner is used for checking the credentials that are provided by the partners. Each state of the choreography contains information about the current state of all the participants (*seller, buyer, credit agency*, and *broker*). For example, the state (**ARTICLE SPECIFIATION SUBMIT)** indicates that the *buyer* is in the **start** state, the *seller* is in the **Sent_Req** state, the *broker* is in the **Rec_Req** state, and the *credit agency* is in the **start** state. The process is based on a *broker* between the *seller* and the *buyer*. The sellers send the articles to the broker and the buyers buy these articles from the broker. Usually, this system is used when the sellers are not specialists and they want to sell some articles.

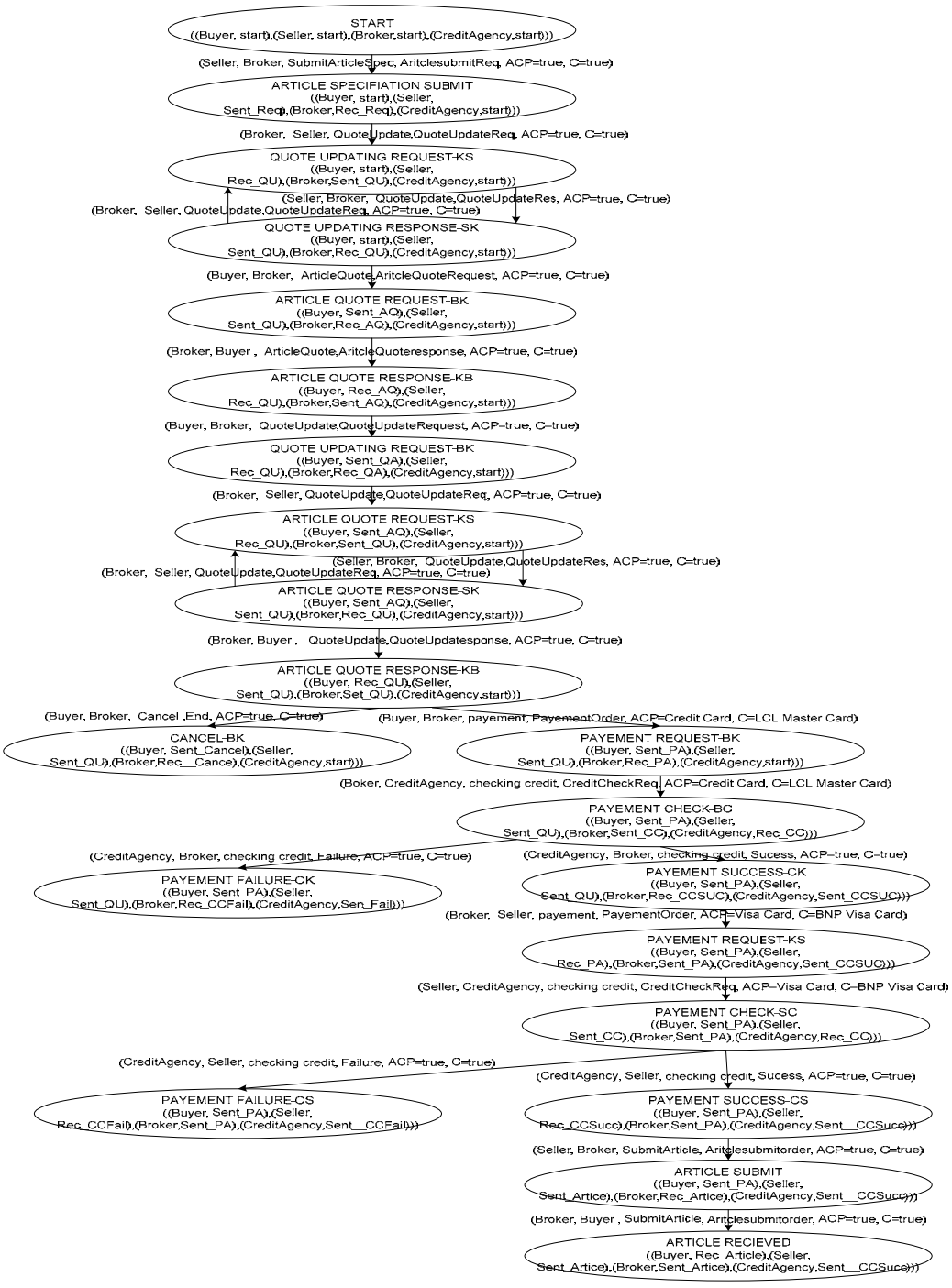

Figure 8-1: Web services choreography describing shopping process

The rest of this section presents the formal definitions of the business protocol and the algorithms that are used in the analysis. The business protocol definition is the same as the previous definition, augmented with the sender or the receiver of each message. This protocol is deterministic (i.e. all the outputs transition from any state are different).

Definition 8-2. (A business protocol assigned with AC) *is a tuple $P = (S; s_0; T; F)$ which consists of the following elements:*

- *S is a finite set of states and $s_0 \in S$ is the initial state.*
- *$T \subseteq S \times S \times M \times ((\{-\} \times 2^c) \cup (\{+\} \times pl)) \times P$, is a finite set of explicit transition, where M is a set of messages assigned to the transitions between the states, pl is the set of access control policies, c is the set of credentials, and P is the protocol which receives or sends the message on this transition.*
- *This protocol is deterministic (i.e. a message cannot correspond to more than one output transition of a state).*
- *All states in the protocol are accessible and co-accessible (i.e. there a path from the start state to this state and from this state to a final state).*
- *$F \subseteq S$ is a set of final states. If $F = \emptyset$ then P is said to be an empty protocol.*

Figure 8-2 shows an example of two business protocols assigned with AC and the message sender or receiver.

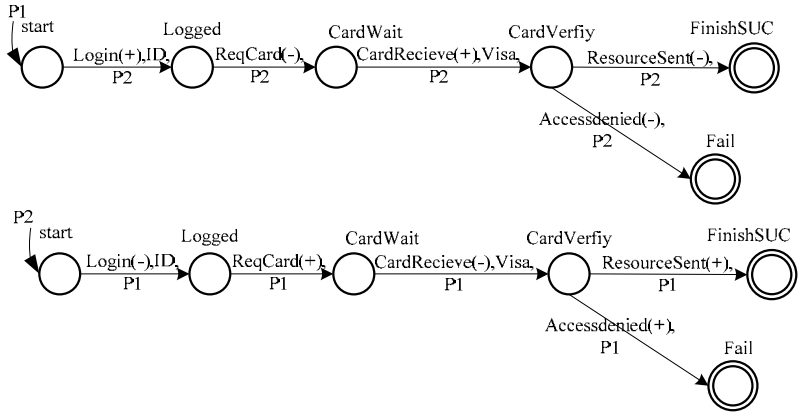

Figure 8-2: Two business protocols assigned with access control and the message sender or receiver.

Each protocol consists of set of states (e.g., **start**, **Logged**, **FinishSUC**) with one start state (**start**) and one or more final states (e.g., **Fail**,**FinishSUC**) and set of transitions. Each transition is guarded by either a message with a positive sign for the messages that will be received (e.g., **Login(+), CardRecieve(+)**) or a message with a negative sign for messages that will be sent (e.g., **ReqCard(-)**, **ResourceSent(-)**). The positive message can be annotated with ACP (e.g., the ID with the message **Login(+)** in the protocol *P1*) and the negative message can be annotated with credentials(e.g., the **Visa** with message **CardRecieve(-)** in the protocol *P2*). The transition is also annotated by the protocol name in which the message will be sent to or received from. The protocol *P1* models a Web service (*W1*) that receives a login message with the *ID* of the service requester (*W2*) which is modeled by the protocol *P2*, then the Web service *W1* sends the **ReqCard** message and waits to receive the **CardRecieve** with **Visa** credential from the Web service *W2* and based on the **Visa** provided, the Web service *W1* can send the **ResourceSent** or the **Accessdenied** message to the Web service *W2*.

Definition 8-3. (The product automata A^p of n business protocols) *$P^1 = (S^1; s_0{}^1; T^1; F^1)$, □, $P^n = (S^n; s_0{}^n; T^n; F^n)$ is defined as $A^p = (S^p; s_0{}^p; T^p; F^p)$ where:*

□ $S^p = S^1 \times □ \times S^n$,

□ $s_0{}^p = (s_0{}^1 ; □ ; s_0{}^n)$.

□ T^p *is the greatest subset of* ($S^p \times S^p \times M \times pl \times 2^c \times P^x \times P^y$) *such that for each transition* (s_i^p; s_{i+1}^p; m_i; pl_i^p ; c_i^p; p^x; p^y) $\in T^p$ *there exist two transitions* (s_i^x; s_{i+1}^x; m_i ; (pl_i^x *or* c_i^x) ;p^y) $\in T^x$ *and* (s_i^y; s_{i+1}^y; m_i ; (pl_i^y *or* c_i^y) ; p^x) $\in T^y$ *with :*

- s_i^x, s_i^y *are in the states which composes* s_i^p *and* s_{i+1}^x, s_{i+1}^y *are in the states which composes* s_{i+1}^p .
- *Polarity*(m_i; T^x) ≠ *polarity*(m_i; T^y) *and*
  - *If polarity*(m_i; P^x) = - *then* pl^x = True, $pl^p{}_i = pl_i^y$, $c_i^p = c_i^x$, c_i^y =*true*
  - *Otherwise* (m_i; P^y) = - *then* pl^y = True, $pl^p{}_i = pl_i^x$, $c_i^p = c_i^y$, c_i^x =*true*

□$F^p = F^1 \times □ \times F^n$.

Figure 8-3 shows an example of two business protocols (*P1* and *P2*) and their product automata *P1* × *P2*. In the product automata model, each state refers to the two corresponding states of the two protocols. The states and the transitions that will never be visited during the probable interactions between the two protocols will not be included in the product automata. For example, the state (**ReqCanceled-1**) in the protocol *P1* will not be included because the message (**CancelReq(+)**) of the *P1* will never be triggered by the protocol *P2*. In terms of the ACP, we present the policy and the credentials on the same common transition (i.e., in the product automata *P1* × *P2*, the transition between the states (**CardWait-1,CardWait-2**) and (**CardVerfiy-1, CardVerfiy-1**) has a (**Visa**) as a policy and a (**MasterCard**) as a credential.

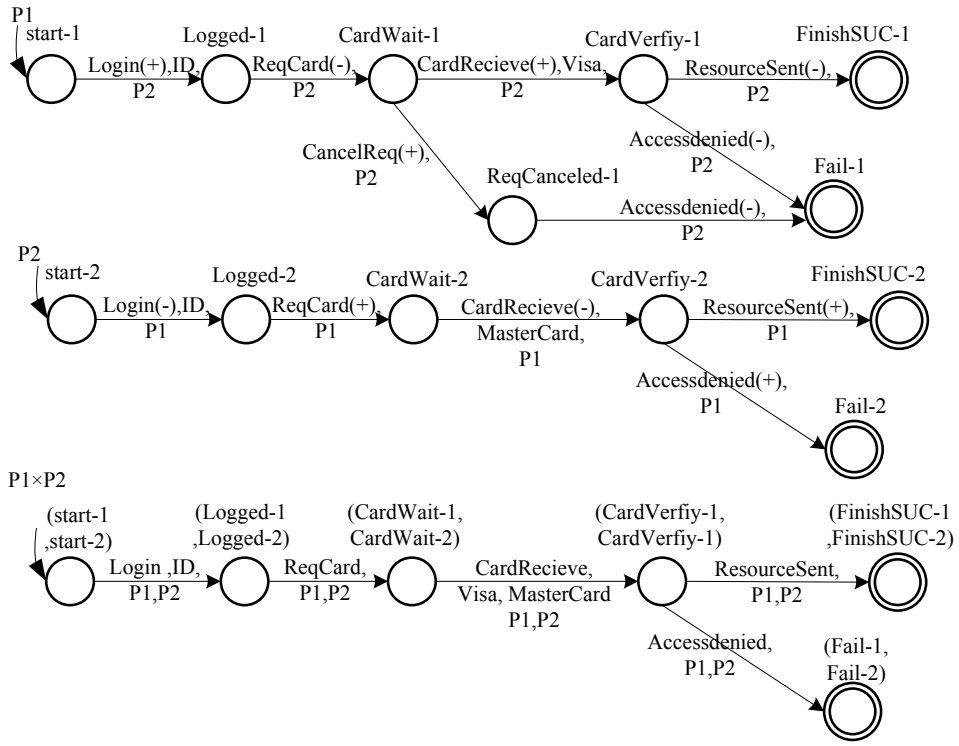

Figure 8-3: Two business protocols (*P1* and *P2*) and their product automata *P1* × *P2*.

Definition 8-4. (Cumulative path in the product automata)*:* $PA^p = s_i^P \xrightarrow{CU_i^{xy}} s_{i+1}{}^p$; □ ; $s_n{}^p \xrightarrow{CU_n^{xy}} s_{n+1}{}^p$ *is a cumulative path in the product automata* $A^p = (S^p; s_0{}^p; T^p; F^p)$ *where*

□ CU_i^{xy} *is the set of cumulative credentials which is received from the protocol* P^x *to the protocol* P^y. CU_i^{xy} *is the union of the previous set of cumulative credentials* CU_{i-1}^{xy} *and the current set of credentials* c_i^{xy} *where* c_i^{xy} *is the set of credentials on the transition between the state* (s_i^x *and* $s_{i+1}{}^x$) *of the protocol* P^x *where* s_i^x *is one of the states that produce* s_i^p *and* $s_{i+1}{}^x$ *is one of the states that produce* $s_{i+1}{}^p$.

□ $CU_0^{xy} = c_0^{xy}$

□*A complete cumulative path in the product automata is the cumulative path which starts with the initial state* s_0^P *and ends with a final state* $s_f^P \in F^p$.

Definition 8-5. *(Co-accessibility of a state in the product of automata):* $A^p = p^1 \times \square \times p^n = (S^p; s_0^p; T^p; F^p)$ *is the product of automata of n BP, state* $s_i^p \in S^p$ *is co-accessible if there exist two paths* PA^1 *and* PA^2 *where* $PA^2 = s_i^p . PA^1 . s_f^p$, *and* $s_f^p \in F^p$.

Definition 8-6. (Compatibility in terms of product automata assigned with AC) *Protocols* $P^1 = (S^1; s_0^1; T^1; F^1), \square, P^n = (S^n; s_0^n; T^n; F^n)$ *and* $A^p = (S^p; s_0^p; T^p; F^p)$ *is their product automata assigned with AC, we say that the n protocols are compatible using their product automata if there is a relation* $R = S^1 \times \square \times S^n$ *where for all* $(s_i^1, \square, s_i^n) \in R$:

$\forall(s_i^x; s_{i+1}^x; m^-; c_i^x; p^y) \in T^x, \exists (s_i^y; s_{i+1}^y; m^+; pl_i^y; p^x) \in T^y$ *where* $(s_i^p; s_{i+1}^p; m; c_i^x; p^x; p_i^y) \in T^p$, *and* $s_{i+1}^p \in R$

$\forall(s_i^y; s_{i+1}^y; m^-; c_i^y; p^x) \in T^y, \exists (s_i^x; s_{i+1}^x; m^+; pl_i^x; p^y) \in T^x$ *where* $(s_i^p; s_{i+1}^p; m; c_i^y; p^y; p_i^x) \in T^p$, *and* $s_{i+1}^p \in R$.

s_i^x, s_i^y *are in the states which composes* s_i^p *and* s_{i+1}^x, s_{i+1}^y *are in the states which composes* s_{i+1}^p.

$s_{i+1}^p \in S^P$ *is co-accessible*

$s_0^p \in R$

□*For all the complete non looping cumulative paths* $PA^p = s_0^P \xrightarrow{cu_0^{xy}} s_1^p$; □ ; $s_n^p \xrightarrow{cu_n^{xy}} s_{n+1}^p$ *, in the product automata, each policy* pl_i^{xy} *is satisfied by the set of cumulative credentials* cu_i^{xy} .

The algorithm which is used for checking the compatibility between *n* protocols in terms of product automata with AC 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 *n* protocols and all the outgoing messages from this state in one protocol can be received by one of the other protocols) then the algorithm stops and the n 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 go to the second part. The second part is for calculating the cumulative credentials on each transition on the product automata.

Algorithm 8-1 presents the second part of the algorithm. The idea of this algorithm is to use the queue data structure to cumulate the credentials. Each element of the queue consists of the state, cumulative credentials which are corresponding to the n protocols in this state. The algorithm traverses through the automata for updating these credentials of the states and in the same time updates the cumulative credentials on the transitions. After calculating the cumulative credentials on each transition, if any ACP related to one of the protocols on any transitions is not satisfied by the cumulative credentials on this transition then the n protocols are not compatible in terms of AC. The AND operator between two credentials in an ACP expression means that these two credentials are required to satisfy this policy and the OR operator between them means that one of these credentials is sufficient for satisfying the policy.

Algorithm 8-1: Compatibility between *n* protocols in terms of AC using cumulative product automata.

Input: $P^1 = (S^1; s_0^1; T^1; F^1)$, □ , $P^n = (S^n; s_0^n; T^n; F^n)$ and $A^p = (S^p; s_0^p; T^p; F^p)$ their product automata $A^p = (S^p; s_0^p; T^p; F^p)$

Output: The n protocols are compatible in terms of ACP or not.

1-Calculate the cumulative credentials on the automata.

CU_i^{xy} : Cumulative credentials sent by protocol P^x to the protocol P^y before reaching to the state s_i^p.

CU_{ij}^{xy} : Cumulative credentials sent by protocol P^x to the protocol P^y and assigned to the transition between s_i^p and s_j^p (i.e. union of set of credentials in those transitions.

For each state $s_i^p \in output(s_0^p)$ **do**

$CU_i^{xy} = CU_0^{xy}$

ENQUEUE(s_i^p; CU_i^{xy})

While Q ≠ empty **do**

Temp_Q = DEQUEUE(Q)

for each $s_j^p \in output(s_i^p)$ in which (s_i^p; CU_i^{xy}) = Temp_Q do

CU_i^{xy} _ temp = CU_i^{xy}

if $CU_j^{xy} \neq$ null then

$CU_i^{xy} = (CU_{ij}^{xy}$ AND $CU_i^{xy})$ OR CU_j^{xy}

else

$CU_i^{xy} = (CU_{ij}^{xy}$ AND $CU_i^{xy})$

$CU_{ij}^{xy} = CU_{ij}^{xy}$ AND CU_i^{xy}

if ¬ ((CU_j^{xy} == CU_{jxy} temp) and $CU_j^{xy} \neq$ null then

ENQUEUE(Q; s_j^p ; CU_1^{xy})

2- **If** $\forall AC_1^{xy}$, $CU_1^{xy} \subseteq AC_1^{xy}$ **in the cumulative product automata satisfying it then**

The n protocols are compatible in terms of ACP.

Else

The n protocols are not compatible in terms of ACP.

Complexity analysis: Let $T_1, ..., T_n$ be the number of transitions of the n protocols $P^1, ..., P^n$ respectively, the construction of the product automata takes (T_1 x T_2 x... x T_n). The calculation of the cumulative credentials will take number of states in the product automata $|S^P|$ multiplied by the size of the longest non looping path which equals the number of states $|S^P|$ multiplied by the maximum number of paths which also equals $|S^P|$ (i.e. cumulative credentials takes $|(S^p)|^3$). As a result, the complexity for the algorithm will be ((T_1 x T_2 x... x T_n) + $|(S^p)|^3$).

8.3. The verification process.

The verification process of Web service selection for choreography implementation based on the compatibility checking with access control can be summarized in the following steps:

1. Select the Web services and get its business protocols assigned with the ACP and credentials.
2. Create the product automata between these protocols (as defined on Definition 8-3).
3. Calculate the cumulative ACC on the product automata (as defined on Definition 8-4).
4. Check the compatibility in terms of AC between these protocols (as defined on Definition 8-6) using algorithm 8-1 for calculating and checking the ACP on the product automata.

5. If the business protocols are compatible in terms of message exchange and AC and the product automata presents the same behaviour as the choreography then the set of services which have these business protocols can implement this choreography. Otherwise, this choreography cannot be implemented by these services.

Example 8-2*:*

Figure 8-4 shows a set of business protocols of Web services for implementing the choreography of Figure 8-1. The protocols are annotated with the AC. Figure 8-5 shows a graphical representation of a simple *Credit Card* ontology that will be used on the verification process. During the compatibility checking algorithm, the provided credentials are checked against the required ACP. The ontology is used during this checking to calculate the subsumption between credentials. For instance in

Figure **8-4**, the *Broker* requires a **Credit Card** policy in the transition (**PayementOrder(+),Credit Card, buyer)** and the corresponding *Buyer* provides **LCL Master Card** credential in the transition (**PayementOrder(-), LCL Master Card, Broker)**. The ontology of Figure 8-5 shows that the **LCL Master Card** is a **Master Card** which is a **Credit Card.** As a consequence, the **LCL Master Card** satisfies the **Credit Card** policy**.** Without using the ontology, the two protocols are not compatible because the checker will not detect the relation between the require policy and the provided credentials.

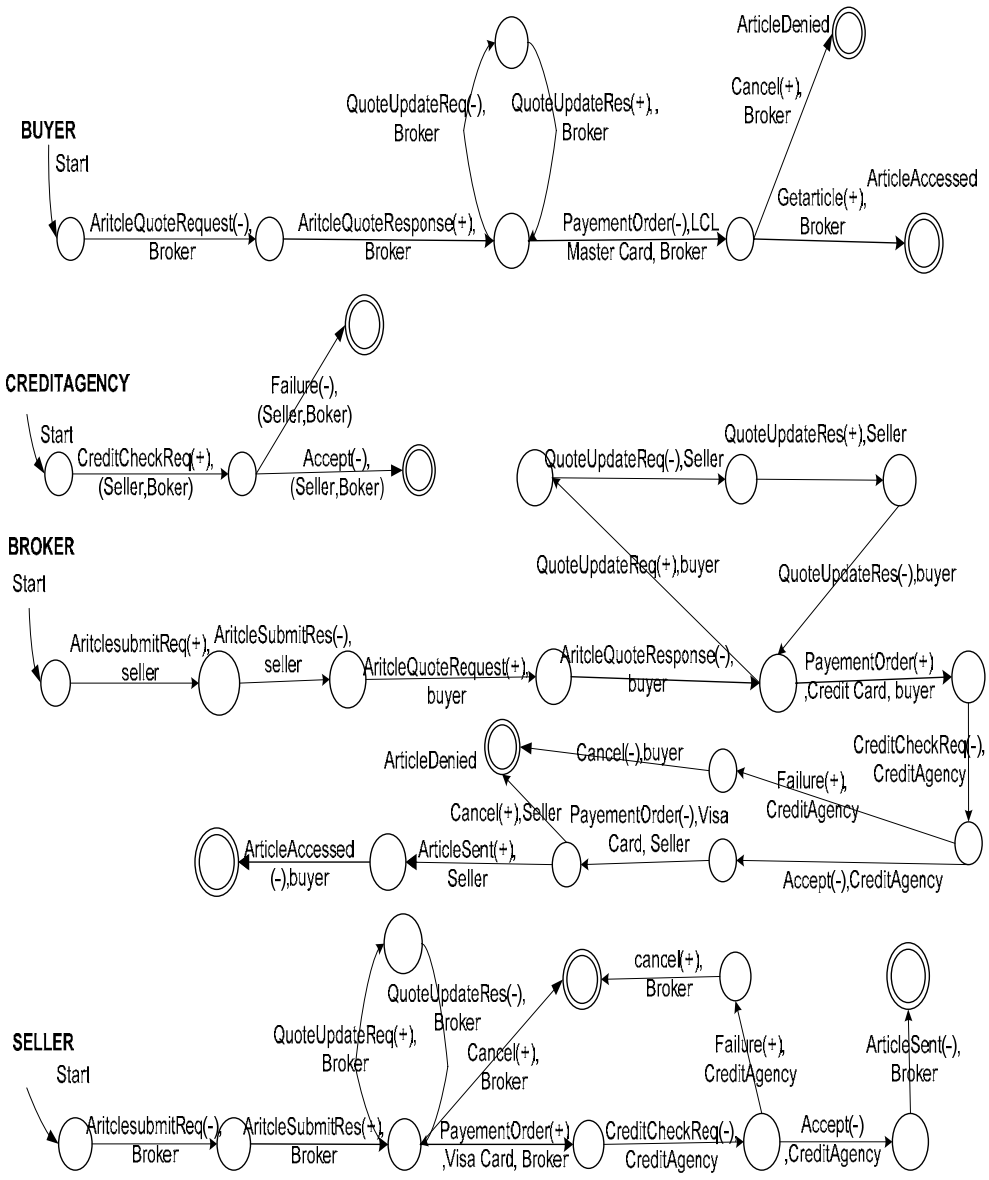

Figure 8-4: Set of business protocols of Web services can be used for implementing the choreography of Figure 8-1.

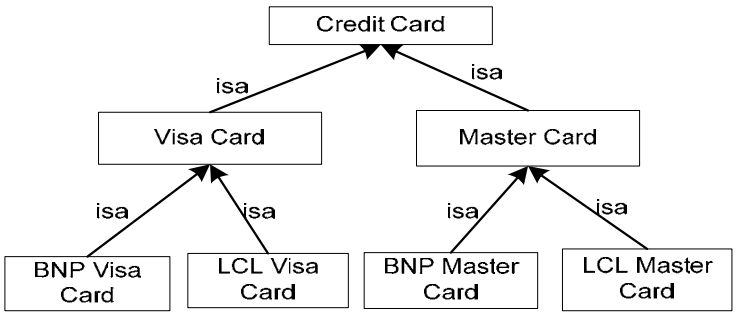

Figure 8-5: Graphical representation of a simple Credit Card ontology which is used in the verification process.

8.4. Related Work

Analyzing services description is presented in many research works. For instance, Fu et al. [72] present a modeling for Web services interactions by formalizing the specification and verification of electronic services for composition purposes. Qiu et al. [116] propose a language for Web services choreographies called Chor as a simplification of WSCDL [87]. Formal analysis of service protocols in terms of automated support to service interoperability at the business protocol level has been discussed in some recent works (e.g. [18, 114, 34, 82, 41, 17, 141, 115]). Foster et al. [8] present a formalization of Web services composition and Web services choreographies based on finite state process algebra. Busi et al. [43] and Kazhamiakin et al. [89] propose two formal calculi to model Web services orchestration and Web services choreography. They investigate the interdependencies between choreography and orchestration and propose a bisimulation-like notion of conformance between choreography and orchestration of Web services. Robinson et al. [119] investigate the problem of how to enforce access control in Web services choreographies. They propose a mechanism to derive access control policies to be enforced by each Web service covering a choreography role and architecture to enforce such policies at runtime. Access control policies enforcement is enabled and disabled in a just-in-time manner that matches the control flow described in the choreography.

Paci et al. [107] present an approach to determine at the design time whether a chorography can be implemented by a set of services based on their access control policies and the disclosed policies regulating the release of their credentials. They check in the design time that all the possible conversations going from the initial

state to the final state in the choreography transition system can be implemented according to the operation access control and disclosed policies of the selected Web services. An equivalent scenario for this verification is to verify set of composite services with access control against set of conversations of business process. They use the idea of assigning AC on business protocol but not for checking the compatibility. In our work, we assign the access control policy on the business protocol of the Web service but for another type of analysis (checking compatibility and replaceability).

Chapter 9 .CONCLUSION AND FUTURE WORK

This chapter concludes the dissertation by summarizing the main results that are achieved by this work. Possible directions for further research and indications for potential applications are given as well.

9.1 Conclusion

The high level interoperability operations analysis such as compatibility and replaceability checking analysis between Web services in the presence of different constraints is the main goal of this thesis. In this respect**,** our research results can be divided to three main parts. The first part deals with modeling and analyzing Web services with time constraints. While doing so, we have proposed a number of new algorithms and formalization. We started by defining the Web service business protocol that presents its behavior and add the time as a part of this description before analysis. Then, the compatibility and replaceability checking definitions are presented. The implicit transitions in the business protocol are one of the problems during the analysis. Therefore, this work presents two algorithms for removing these transitions before performing the analysis. The first algorithm works with the one-clock business protocols and the second algorithm works with the multi-clocks business protocols (chapter 3 and chapter 4).

The second part deals with the Web services modeling and analyzing after annotating the ACP (chapter 5). Web services analysis after annotating the AC emerging the problem of the ACP cumulating. We have discussed this problem and presented the solution by calculating the cumulative AC on each transition after creating the product automata. The research in part is extended to include the implementation of a complex business processes by selecting compatible set of Web services in terms of AC and in terms of message exchanged (chapter 8). In order to achieve this task, the definition of the business protocol is extended to attach each message on each transition with a parameter indicates the sender or receiver of it. We present an approach for verifying that a set of Web services are compatible and im-

plementing choreography of a complex business process in terms of message specifications and AC.

The third part is the message specification approach which tries to give a general approach for modeling and analyzing the different types of constraints. This approach tries to categorize these constraints and presents for each category the suitable manipulation. For example, the constraint that needs cumulating such as the AC is one type of these categories. The adaptive compatibility approach is provided in conjunction with the general specification approach. The adaptive compatibility approach tries to avoid the undecidability problem for complex message specifications constraints. It gives the consumer a wide range of choices and balance between the required properties that are needed for checking and the complexity that can be accepted. The consumer can decide based on his Web service the way in which the compatibility checking process takes. For instance, we can check all the properties in the same time because we have a powerful machine or the Web services have not complex specifications. In another situation, we can check first the message Schema attributes and then if it is compatible we check the time and if it is compatible we check the AC and continue until checking all the specifications or halt due to incompatible attributes.

9.2 Future work

As the work for this thesis progressed, a number of areas deserving further research revealed themselves. In particular, this work can be extended in four axes. The first axis is the trust negotiation approaches between Web services based on compatibility and replaceability checking. Trust negotiation consists of a bilateral disclosure of digital credentials and the trust is built incrementally by disclosing these credentials according to the disclosure policies. Therefore; compatibility checking before starting the disclosing of the credentials is an important step in the trust negotiation process. In addition, the replaceability is important in case of the incompatible Web service in terms of AC. For instance, the consumer or the service provider could replace the services by others to achieve the trust negotiation process if this does not disrupt the main functions. As a result, we will try to make a link between the trust negotiation policy languages and the compatibility and replaceability checking approach in order to enable the proposed integration.

The second axis is extending our work to include the time constraints and the general specifications constraints in selecting Web services for choreography imple-

mentation using compatibility checking approach. In terms of time constraints, we will first remove the implicit transitions. The problem in compatibility checking is the need for synchronization in not only for each two protocol but also for more than one protocol in the same time. This situation is resulted because each Web service can interact with more than one service in the process. Therefore, an intelligent approach is needed to overcome this problem and the other problems which can be emerged after assigning the message specification constraints.

The Third axis is the development of adaptors based on the incompatible properties between web services. The adaptors will be different if each adaptor solves one type of incompatibility. For instance, the adaptor for the incompatibility due to the message order is different than the adaptor for AC or trust negotiation adaptor. The problem with the adaptors is that the standardization of the Web service is emerged to avoid adaptors. Therefore, there is a need for general adaptors not specific one. These adaptors can be reused in solving all the incompatibility problems.

The Fourth axis is the development of a Web service behavior specification language represents the Web service business protocols. This language expresses all the types of specification and constraints that can be attached with the service. This is a hope and I propose to adopt it as a big international project because this language will deal with a very large scale of data domains and many research directions. For instance, this language should express the time constraints, AC, the message meaning, privacy, and other constraints. This language can help in many directions and can be passed in the standardization process to make use of this standardization in the future analysis. As a result, the Web service behavior description is represented as a new standard part of the Web service.

9.3 Publications

International refereed conferences

- Emad Elabd, Emmanuel Coquery, Mohand-Said Hacid. **Timed Web services analysis after removing complex implicit transitions** In the IEEE International Conference on Web Services (ICWS) (Acceptance Rate 14%), IEEE ed. Washington D.C, USA. 2011.
- Emad Elabd, Emmanuel Coquery, Mohand-said Hacid . **Selecting Web Services for Choreography Implementation: Compatibility Checking Approach with Access Control**. In the 22nd Interna-

tional Conference on Software Engineering and Knowledge Engineering(Acceptance Rate 33.0%) , Knowledge Systems Institute Graduate School ed. San Francisco Bay, USA. 2010.

- Emad Elabd, Emmanuel Coquery, Mohand-Said Hacid. **Checking Compatibility and Replaceability in Web Services Business Protocols with Access Control.** In the IEEE International Conference on Web Services (ICWS) (Acceptance Rate 17.5%), IEEE ed. Miami,Florida,USA. pp. 409-416.
- Emad Elabd, Emmanuel Coquery, Mohand-Said Hacid. **Compatibility and Replaceability Analysis of Timed Web Services Protocols**. In the ICCEE, IEEE Computer Society ed. Dubai, UAE. pp. 15-19. ISBN 978-0-7695-392. 2009.

National refereed conferences

- Emad Elabd, Emmanuel Coquery, Mohand-said Hacid, Sélection de services Web pour l'implémentation de chorégraphies: vérification de compatibilité avec contrôles d'accès. In the 26èmes journées Bases de Données Avancées,Toulouse, France. 2010.

Technical Reports and demonstration

- Emad Elabd, Emmanuel Coquery, Mohand-said Hacid. **Checking Compatibility and Replaceability inWeb Services Business Protocols with Access Control. E. Elabd, E Coquery, M. Hacid**. Rapport de recherche RR-LIRIS-2009-030 2009.

- COMPAS project the second review in Brussels
- Poster in the third day of theses in Lyon

BIBLIOGRAPHY

[1] Serge Abiteboul, Victor Vianu, Brad Fordham, and Yelena Yesha. Relational transducers for electronic commerce. In *Proceedings of the seventeenth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems*, PODS '98, pages 179–187, New York, NY, USA, 1998. ACM.

[2] Serge Abiteboul, Victor Vianu, Brad Fordham, and Yelena Yesha. Relational transducers for electronic commerce. *Journal of Computer and System Sciences*, 61:236–269, October 2000.

[3] Vikas Agarwal, Girish Chafle, Koustuv Dasgupta, Neeran Karnik, Arun Kumar, Sumit Mittal, and Biplav Srivastava. Synthy: A system for end to end composition of web services. *Web Semantics: Science, Services and Agents on the World Wide Web*, 3(4):311 – 339, 2005. World Wide Web Conference 2005-Semantic Web Track.

[4] Gustavo Alonso, Fabio Casati, Harumi A. Kuno, and Vijay Machiraju. *Web Services-Concepts, Architectures and Applications*. Data-Centric Systems and Applications. Springer, 2004.

[5] Rajeev Alur and David L. Dill. Automata for modeling real-time systems. In *Proceedings of the seventeenth international colloquium on Automata, languages and programming*, pages 322–335, New York, NY, USA, 1990. Springer-Verlag New York, Inc.

[6] Alexandre Alves, Assaf Arkin, Sid Askary, Ben Bloch, Francisco Curbera, Yaron Goland, Neelakantan Kartha, Sterling, Dieter König, Vinkesh Mehta, Satish Thatte, Danny van der Rijn, Prasad Yendluri, and Alex Yiu. Web services business process execution language version 2.0. OASIS Committee Draft, May 2006.

[7] Claudio Agostino Ardagna, Ernesto Damiani, Sabrina De Capitani di Vimercati, and Pierangela Samarati. A web service architecture for enforcing access control policies. *Electronic Notes in Theoretical Computer Science*, 142:47–62, 2006.

[8] Ali Arsanjani. Introduction to special issue on developing and integrating enterprise components and services. *Communications of the ACM*, 45(10):30–34, 2002.

[9] Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, 2003.

[10] Siddharth Bajaj, Don Box, Dave Chappell, and al. Web services policy 1.2 - framework (ws-policy). Technical report, BEA Systems Inc, http://www.w3.org/Submission/WS-Policy/, 25 April 2006.

[11] Siddharth Bajaj, Don Box, Dave Chappell, Francisco Curbera, Glen Daniels, Phillip Hallam-Baker, Maryann Hondo, Chris Kaler, and etc. Web services policy framework (ws-policy), "http://www.verisign.com/corporate/research-ws-policy.pdf", 2006.

[12] Karim Baïna, Boualem Benatallah, Fabio Casati, and Farouk Toumani. Model-driven web service development. In *Advanced Information Systems Engineering*, volume 3084 of *Lecture Notes in Computer Science*, pages 527–543. Springer Berlin / Heidelberg, 2004. 10.1007/978-3-540-25975-6_22.

[13] A. Banerji, C. Bartolini, D. Beringer, V. Chopella, and et. Web services conversation language (wscl) 1.0, http://www.w3.org/tr/wscl10/. Technical report, March 2002.

[14] Chitta Baral and Michael Gelfond. Logic programming and knowledge representation. *Journal of Logic Programming*, 19:73–148, 1994.

[15] Tom Bellwood, Luc Clément, David Ehnebuske, Andrew Hately, Maryann Hondo, and et al. "uddi version 3.0": Universal description, discovery and integration (uddi) project, published specification,"http://uddi.xml.org/", "http://www.uddi.org/pubs/uddi_executive_white_paper.pdf", July 2002.

[16] Boualem Benatallah, Fabio Casati, Julien Ponge, and Farouk Toumani. Compatibility and replaceability analysis for timed web service protocols. In *Bases de Données Avancées(BDA)*, 2005.

[17] Boualem Benatallah, Fabio Casati, Julien Ponge, and Farouk Toumani. On temporal abstractions of web service protocols. In *International Conference on Advanced Information Systems Engineering (CAiSE) Short Paper Proceedings*, 2005.

[18] Boualem Benatallah, Fabio Casati, and Farouk Toumani. Analysis and management of web service protocols. In *International conference on conceptual modeling (ER)*, pages 524–541, 2004.

[19] Boualem Benatallah, Fabio Casati, and Farouk Toumani. Web service conversation modeling: A cornerstone for e-business automation. *IEEE Internet Computing*, 8(1):46–54, 2004.

[20] Boualem Benatallah, Fabio Casati, and Farouk Toumani. Representing, analysing and managing web service protocols. *Data & Knowledge Engineering*, 58(3):327–357, 2006.

[21] Boualem Benatallah, Fabio Casati, Farouk Toumani, Julien Ponge, and Hamid R. Motahari Nezhad. Service mosaic: A model-driven frame-

work for web services life-cycle management. *IEEE Internet Computing*, 10(4):55–63, 2006.

[22] Daniela Berardi, Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, and Massimo Mecella. Automatic service composition based on behavioral descriptions. *International Journal of Cooperative Information Systems*, 14(4):333–376, 2005.

[23] Daniela Berardi, Fabio De Rosa, Luca De Santis, and Massimo Mecella. Finite state automata as conceptual model for e-services. *Journal of Integrated Design & Process Science*, 8(2):105–121, 2004.

[24] Tim Berners-Lee. Artificial intelligence and the semantic web: Aaai2006 keynote. w3c web site 2006. url: http://www.w3.org/2006/talks/0718-aaai-tbl/overview.html. lastaccessed 27/09/2010.

[25] Tim Berners-Lee. Semantic web - xml2000. w3c web site 2000, url: http://www.w3.org/2000/talks/1206-xml2k-tbl/slide10-0.html. last accessed 27/09/2010.

[26] Tim Berners-Lee. The semantic web and challenges. w3c web site slideshow 2003. url: http://www.w3.org/2003/talks/01-sweb-tbl/slide15-0.html. lastaccessed 27/09/2010.

[27] Tim Berners-Lee. Www past and future. w3c web site 2003. url: http://www.w3.org/2003/talks/0922-rsoc-tbl/slide30-0.html. lastaccessed 27/09/2010.

[28] Tim Berners-Lee, James Hendler, and Ora Lassila. The Semantic Web. *Scientific American*, 284(5):34–43, 2001.

[29] Elisa Bertino, Anna C. Squicciarini, Ivan Paloscia, and Lorenzo Martino. Ws-ac: A fine grained access control system for web services. *World Wide Web*, 9(2):143–171, 2006.

[30] Elisa Bertino, Anna Cinzia Squicciarini, Lorenzo Martino, and Federica Paci. An adaptive access control model for web services. *International Journal of Web Services Research (JWSR)*, 3(3):27–60, 2006.

[31] Claudio Bettini, Xiaoyang Sean Wang, and Sushil Jajodia. Temporal reasoning in workflow systems. *Distributed and Parallel Databases*, 11(3):269–306, 2002.

[32] Dirk Beyer, Arindam Chakrabarti, and Thomas A. Henzinger. Web service interfaces. In *Proceedings of the 14th ACM International World Wide Web Conference (WWW 2005, Chiba, May 10-14)*, pages 148–159. ACM Press, New York (NY), 2005.

[33] Mikolaj Bojanczyk, Anca Muscholl, Thomas Schwentick, Luc Segoufin, and Claire David. Two-variable logic on words with data. In *Proceedings of the 21st Annual IEEE Symposium on Logic in Computer Science*, pages 7–16, Washington, DC, USA, 2006. IEEE Computer Society.

[34] Lucas Bordeaux, Gwen Salaün, Daniela Berardi, and Massimo Mecella. When are two web services compatible? In *Technologies for E-Services*, volume 3324 of *Lecture Notes in Computer Science*, pages 15–28. Springer Berlin / Heidelberg, 2005. 10.1007/978-3-540-31811-8_2.

[35] Ahmed Bouajjani, Cezara Dragoi, Constantin Enea, Yan Jurski, and Mihaela Sighireanu. A generic framework for reasoning about dynamic networks of infinite-state processes. *Logical Methods in Computer Science*, 5(2), 2009.

[36] Ahmed Bouajjani, Peter Habermehl, Yan Jurski, and Mihaela Sighireanu. Rewriting systems with data. In *Proceedings of the 16th international symposium on Fundamentals of Computation Theory*, pages 1–22, Berlin, Heidelberg, 2007. Springer-Verlag.

[37] Patricia Bouyer. A logical characterization of data languages. *Information Processing Letters*, 84(2):75–85, 2002.

[38] Patricia Bouyer, Antoine Petit, and Denis Thérien. An algebraic approach to data languages and timed languages. *Information and Computation*, 182:137–162, May 2003.

[39] Daniel Brand and Pitro Zafiropulo. On communicating finite-state machines. *Journal of the ACM*, 30(2):323–342, April 1983.

[40] Tevfik Bultan. Modeling interactions of web software. In *Proceedings of the 2nd Int'l. Workshop on Automated Specification and Verification of Web Systems*, pages 45–52, Washington, DC, USA, 2006. IEEE Computer Society.

[41] Tevfik Bultan, Xiang Fu, Richard Hull, and Jianwen Su. Conversation specification: a new approach to design and analysis of e-service composition. In *WWW '03: Proceedings of the 12th international conference on World Wide Web*, pages 403–410, New York, NY, USA, 2003. ACM.

[42] Tevfik Bultan, Jianwen Su, and Xiang Fu. Analyzing conversations of web services. *IEEE Internet Computing*, 10(1):18–25, 2006.

[43] Nadia Busi, Roberto Gorrieri, Claudio Guidi, Roberto Lucchi, and Gianluigi Zavattaro. Choreography and orchestration conformance for system design. In *In COORDINATION, volume 4038 of Lecture Notes in Computer Science*, pages 63–81. Springer, 2006.

[44] Marco Carbone, Kohei Honda, and Nobuko Yoshida. A calculus of global interaction based on session types. *Electronic Notes in Theoretical Computer Science*, 171(3):127–151, 2007.

[45] Jorge Cardoso. *Semantic Web Services: Theory, Tools and Applications*. IGI Global, 2007.

[46] Arindam Chakrabarti, Luca de Alfaro, Thomas A. Henzinger, and Mariïlle Stoelinga. Resource interfaces. In *Embedded Soft-*

ware(EMSOFT), volume 2855 of *Lecture Notes in Computer Science*, pages 117–133. Springer, 2003.

[47] Fahima Cheikh, Giuseppe De Giacomo, and Massimo Mecella. Automatic web services composition in trustaware communities. In *SWS '06: Proceedings of the 3rd ACM workshop on Secure web services*, pages 43–52, New York, NY, USA, 2006. ACM.

[48] E. Christensen, F. Curbera, G. Meredith, and S. Weerawarana. Web services description language (WSDL) 1.1, http://www.w3.org/tr/wsdl. W3c note, March 2001.

[49] Lawrence Chung. Representation and utilization of non-functional requirements for information system design. In *In Proceedings of the 3rd International Conference on Advanced Information Systems Engineering - CAiSE'91, April 7-11, 1991 Trodheim, Norway, LNCS, pages 5–30. Springer-Verlag.*, pages 5–30.

[50] Marijke Coetzee and Jan H. P. Eloff. A trust and context aware access control model for web services conversations. In *Trust, Privacy and Security in Digital Business*, volume 4657 of *Lecture Notes in Computer Science*, pages 115–124. Springer, 2007.

[51] P. Cousot and R. Cousot. Comparing the Galois connection and widening/narrowing approaches to abstract interpretation, invited paper. In *Proceedings of the International Workshop Programming Language Implementation and Logic Programming, PLILP'92,*, Leuven, Belgium, 13–17 August 1992, Lecture Notes in Computer Science 631, pages 269–295. Springer-Verlag, Berlin, Germany, 1992.

[52] Patrick Cousot and Radhia Cousot. Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints. In *Proceedings of the 4th ACM SIGACT-SIGPLAN symposium on Principles of programming languages*, POPL '77, pages 238–252, New York, NY, USA, 1977. ACM.

[53] Nicodemos Damianou, Naranker Dulay, Emil Lupu, and Morris Sloman. The ponder policy specification language. In *POLICY '01 Proceedings of the International Workshop on Policies for Distributed Systems and Networks*, volume 1995 of *Lecture Notes in Computer Science*, pages 18–38. Springer, 2001.

[54] Gregorio Dïaz, Juan José Pardo, Marïa-Emilia Cambronero, Valentin Valero, and Fernando Cuartero. Verification of web services with timed automata. *Electronic Notes in Theoretical Computer Science*, 157(2):19–34, 2006.

[55] Luca de Alfaro and Thomas A. Henzinger. Interface automata. In *ESEC/FSE-9 Proceedings of the 8th European software engineering conference held jointly with 9th ACM SIGSOFT international symposium on Foundations of software engineering*, pages 109–120, 2001.

[56] Luca de Alfaro, Thomas A. Henzinger, and Mariï¿½lle Stoelinga. Timed interfaces. In *Proceedings of the Second International Conference on Embedded Software(EMSOFT)*, volume 2491 of *Lecture Notes in Computer Science*, pages 108–122. Springer, 2002.

[57] Stephane Demri and Ranko Lazic. Ltl with the freeze quantifier and register automata. In *Proceedings of the 21st Annual IEEE Symposium on Logic in Computer Science*, pages 17–26, Washington, DC, USA, 2006. IEEE Computer Society.

[58] Stéphane Demri, Ranko Lazic, and Arnaud Sangnier. Model checking freeze ltl over one-counter automata. Ín *Proceedings of the Theory and practice of software, 11th international conference on Foundations of software science and computational structures*, FOSSACS'08/ETAPS'08, pages 490–504, Berlin, Heidelberg, 2008. Springer-Verlag.

[59] Alin Deutsch, Liying Sui, and Victor Vianu. Specification and verification of data-driven web applications. *Journal of Computer and System Sciences*, 73:442–474, May 2007.

[60] Arulazi Dhesiaseelan and Venkatavaradan Ragunathan. Web services container reference architecture (wscra). In *ICWS '04: Proceedings of the IEEE International Conference on Web Services*, page 806, Washington, DC, USA, 2004. IEEE Computer Society.

[61] Marlon Dumas, Boualem Benatallah, and Hamid R. Motahari Nezhad. Web service protocols: Compatibility and adaptation. *IEEE Data Engineering Bulletin*, 31(3):40–44, 2008.

[62] Gregorio Dyaz, M. Emilia Cambronero, Juan J. Pardo, Valentin Valero, and Fernando Cuartero. Automatic generation of correct web services choreographies and orchestrations with model checking techniques. In *AICT/ICIW*, page 186. IEEE Computer Society, 2006.

[63] Emad Elabd, Emmanuel Coquery, and Mohand-Said Hacid. Checking compatibility and replaceability in web services business protocols with access control. In *Proceedings of the IEEE International Conference on Web Services (ICWS)*, pages 409–416, 2010.

[64] Emad Elabd, Emmanuel Coquery, and Mohand-Said Hacid. Selecting web services for choreography implementation: Compatibility checking approach with access control. In *SEKE*, 2010.

[65] Emad Elabd, Emmanuel Coquery, and Mohand-Said Hacid. Timed web services analysis after removing complex implicit transitions. In *Proceedings of the IEEE International Conference on Web Services (ICWS)*, pages 409–416, 2011.

[66] Dieter Fensel, Wolfgang Wahlster, and Henry Lieberman, editors. *Spinning the Semantic Web: Bringing the World Wide Web to Its Full Potential*. MIT Press, Cambridge, MA, USA, 2002.

[67] David F. Ferraiolo, Ravi Sandhu, Serban Gavrila, D. Richard Kuhn, and Ramaswamy Chandramouli. Proposed nist standard for role-based access control. *ACM Transactions on Information and System Security (TISSEC)*, 4(3):224–274, 2001.

[68] Wan Fokkink. *Introduction to Process Algebra*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2000.

[69] Charles Forgy. Rete: A fast algorithm for the many pattern/many object pattern match problem. *Artificial Intelligences*, 19(1):17–37, 1982.

[70] Charles Lanny Forgy. *On the efficient implementation of production systems.* PhD thesis, Pittsburgh, PA, USA, 1979.

[71] Xiang Fu, Tevfik Bultan, and Jianwen Su. Analysis of interacting bpel web services. In Stuart I. Feldman, Mike Uretsky, Marc Najork, and Craig E. Wills, editors, *WWW*, pages 621–630. ACM, 2004.

[72] Xiang Fu, Tevfik Bultan, and Jianwen Su. Conversation protocols: A formalism for specification and verification of reactive electronic services. In *In Proc. Int. Conf. on Implementation and Application of Automata (CIAA*, pages 188–200. Springer, 2004.

[73] Simon Gay, Malcolm Hole, and Surrey Tw Ex. Types for correct communication in client-server systems. Technical report, Department of Computer Science, Royal Holloway, University of London, 2000.

[74] Simon J. Gay and Malcolm Hole. Types and subtypes for client-server interactions. In S. Doaitse Swierstra, editor, *ESOP*, volume 1576 of *Lecture Notes in Computer Science*, pages 74–90. Springer, 1999.

[75] Aurona Gerber, Alta van der Merwe, and Andries Barnard. A functional semantic web architecture. In *ESWC'08: Proceedings of the 5th European semantic web conference on The semantic web*, pages 273–287, Berlin, Heidelberg, 2008. Springer-Verlag.

[76] Arthur Gill. *Introduction to the Theory of Finite-State Machines*. McGraw Hill, 1962.

[77] Thomas R. Gruber. Toward principles for the design of ontologies used for knowledge sharing? *International Journal of Human-Computer Studies*, 43(5-6):907–928, 1995.

[78] Nawal Guermouche, Olivier Perrin, and Christophe Ringeissen. Timed specification for web services compatibility analysis. *Electronic Notes in Theoretical Computer Science*, 200(3):155–170, 2008.

[79] Rachid Hamadi and Boualem Benatallah. A petri net-based model for web service composition. In *ADC '03: Proceedings of the 14th Australasian database conference*, pages 191–200, Darlinghurst, Australia, Australia, 2003. Australian Computer Society, Inc.

[80] Thomas Hardjono and Nathan Klingenstein. OASIS Security Services (SAML) TC, http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=security.

[81] Kohei Honda, Vasco Thudichum Vasconcelos, and Makoto Kubo. Language primitives and type discipline for structured communication-based programming. In *ESOP '98: Proceedings of the 7th European Symposium on Programming*, pages 122–138, London, UK, 1998. Springer-Verlag.

[82] Richard Hull, Michael Benedikt, Vassilis Christophides, and Jianwen Su. E-services: a look behind the curtain. In *PODS '03: Proceedings of the twenty-second ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 1–14, New York, NY, USA, 2003. ACM.

[83] San-Yih Hwang, Chuan Yin, and Chien-Hsiang Lee. Selecting web services and participants for enforcing workflow access control. In *HICSS*, pages 1–10. IEEE Computer Society, 2009.

[84] Marcin Jurdzinski and Ranko Lazic. Alternation-free modal mu-calculus for data trees. In *Proceedings of the 22nd Annual IEEE Symposium on Logic in Computer Science*, pages 131–140, Washington, DC, USA, 2007. IEEE Computer Society.

[85] Lalana Kagal, Tim Finin, and Anupam Joshi. A policy language for a pervasive computing environment. In *POLICY '03: Proceedings of the 4th IEEE International Workshop on Policies for Distributed Systems and Networks*, page 63, Washington, DC, USA, 2003. IEEE Computer Society.

[86] Zuling Kang, Hongbing Wang, and Patrick C. K. Hung. Ws-cdl+: An extended ws-cdl execution engine for web service collaboration. In *Proceedings of the IEEE International Conference on Web Services (ICWS)*, pages 928–935. IEEE Computer Society, 2007.

[87] Nickolas Kavantzas, David Burdett, Gregory Ritzinger, Tony Fletcher, and Yves Lafon. Web services choreography description language version 1.0 (wscdl). Technical report, W3C(MIT, ERCIM, Keio), http://-www.w3.org/TR/ws-cdl-10/, 2005.

[88] Raman Kazhamiakin, Paritosh K. Pandya, and Marco Pistore. Timed modelling and analysis in web service compositions. In *ARES*, pages 840–846, 2006.

[89] Raman Kazhamiakin and Marco Pistore. Choreography conformance analysis: Asynchronous communications and information alignment. In *WS-FM*, pages 227–241, 2006.

[90] M. Kifer, G. Lausen, and J. Wu. Logical foundations of object-oriented and frame-based languages. *Journal of the ACM*, 42:741–843, 1995.

[91] Heather Kreger. Fulfilling the web services promise. *Communications of the ACM*, 46(6):29–ff, 2003.

[92] Kim G. Larsen, Paul Pettersson, and Wang Yi. Uppaal in a nutshell. *International Journal on Software Tools for Technology Transfer (STTT)*, 1:134–152, 1997. 10.1007/s100090050010.

[93] Ranko Lazic, Tom Newcomb, Joël Ouaknine, A. W. Roscoe, and James Worrell. Nets with tokens which carry data. In *Proceedings of the 28th international conference on Applications and theory of Petri nets and other models of concurrency*, ICATPN'07, pages 301–320, Berlin, Heidelberg, 2007. Springer-Verlag.

[94] Hector Levesque, Fiora Pirri, and Ray Reiter. Foundations for the situation calculus, 1998.

[95] Ninghui Li, John C. Mitchell, and William H. Winsborough. Design of a role-based trust-management framework. In *IEEE Symposium on Security and Privacy*, pages 114–130, 2002.

[96] Michele Mancioppi, Manuel Carro, Willem-Jan van den Heuvel, and Mike P. Papazoglou. Sound multi-party business protocols for service networks. In *ICSOC*, pages 302–316, 2008.

[97] Elisabetta De Maria, Angelo Montanari, and Marco Zantoni. An automaton-based approach to the verification of timed workflow schemas. In *TIME*, pages 87–94. IEEE Computer Society, 2006.

[98] Francis McCabe, David Booth, Hugo Haas, Eric Newcomer, Michael Champion, Chris Ferris, and David Orchard. Web services architecture –w3c working group note. http://www.w3.org/tr/ws-arch/, February 2004.

[99] Sheila A. Mcilraith, Tran Cao Son, and Honglei Zeng. Semantic web services. *IEEE Intelligent Systems*, 16:46–53, 2001.

[100] Massimo Mecella, Mourad Ouzzani, Federica Paci, and Elisa Bertino. Access control enforcement for conversation-based web services. In *WWW '06: Proceedings of the 15th international conference on World Wide Web*, pages 257–266, New York, NY, USA, 2006. ACM.

[101] Tim Moses. extensible access control markup language (xacml) version 2.0. Technical report, OASIS Standard, http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-core-spec-os.pdf, Feb 2005.

[102] Dimitris Mostrous and Nobuko Yoshida. Two session typing systems for higher-order mobile processes. In *TLCA'07: Proceedings of the 8th international conference on Typed lambda calculi and applications*, pages 321–335, Berlin, Heidelberg, 2007. Springer-Verlag.

[103] Wolfgang Nejdl, Daniel Olmedilla, Marianne Winslett, and Charles C. Zhang. Ontology-based policy specification and management. In *ESWC*, pages 290–302, 2005.

[104] Frank Neven, Thomas Schwentick, and Victor Vianu. Finite state machines for strings over infinite alphabets. *ACM Transactions on Computational Logic (TOCL)*, 5:403–435, July 2004.

[105] Hamid R. Motahari Nezhad, Boualem Benatallah, Fabio Casati, and Farouk Toumani. Web services interoperability specifications. *Computer*, 39(5):24–32, 2006.

[106] Federica Paci, Elisa Bertino, and Jason Crampton. An access-control framework for ws-bpel. *International Journal of Web Services Research (JWSR)*, 5(3):20–43, 2008.

[107] Federica Paci, Mourad Ouzzani, and Massimo Mecella. Verification of access control requirements in web services choreography. In *SCC '08: Proceedings of the 2008 IEEE International Conference on Services Computing*, pages 5–12, Washington, DC, USA, 2008. IEEE Computer Society.

[108] Mike. P. Papazoglou and D. Georgakopoulos. Introduction. *Communications of the ACM*, 46(10):24–28, 2003.

[109] Mike P. Papazoglou, Paolo Traverso, Schahram Dustdar, and Frank Leymann. Service-oriented computing: State of the art and research challenges. *IEEE Computer*, 40(11):38–45, 2007.

[110] Mike P. Papazoglou and Willem-Jan van den Heuvel. Service oriented architectures: approaches, technologies and research issues. *The International Journal on Very Large Data Bases*, 16(3):389–415, 2007.

[111] Shamimabi Paurobally and Nicholas R. Jennings. Protocol engineering for web services conversations. *Engineering Applications of Artificial Intelligence*, 18(2):237–254, 2005.

[112] A. Pnueli. The temporal logic of programs. In *18th Annual Symposium on the Foundations of Computer Science (FOCS-77)*, pages 46–57, Providence, Rhode Island, October MarchJanuary–November February 1977. IEEE, IEEE Computer Society Press.

[113] Julien Ponge. *Model Based Analysis of Time-aware Web Services Interactions*. PhD thesis, PhD Thesis, July 2008.

[114] Julien Ponge, Boualem Benatallah, Fabio Casati, and Farouk Toumani. Fine-grained compatibility and replaceability analysis of timed web service protocols. In *ER*, pages 599–614, 2007.

[115] Shankar R. Ponnekanti and Armando Fox. Interoperability among independently evolving web services. In *Middleware '04: Proceedings of the 5th ACM/IFIP/USENIX international conference on Middleware*, pages 331–351, New York, NY, USA, 2004. Springer-Verlag New York, Inc.

[116] Zongyan Qiu, Xiangpeng Zhao, Chao Cai, and Hongli Yang. Towards the theoretical foundation of choreography. In *WWW*, pages 973–982, 2007.

[117] Pemadeep Ramsokul and Arcot Sowmya. An adaptable formal model for web services protocols. In *ICIW '07: Proceedings of the Second International Conference on Internet and Web Applications and Services*, page 40, Washington, DC, USA, 2007. IEEE Computer Society.

[118] Wolfgang Reisig. *Petri Nets: An Introduction*, volume 4 of *Monographs in Theoretical Computer Science. An EATCS Series*. Springer, 1985.

[119] Philip Robinson, Florian Kerschbaum, and Andreas Schaad. From business process choreography to authorization policies. In Ernesto Damiani and Peng Liu, editors, *Data and Applications Security XX*, volume 4127 of *Lecture Notes in Computer Science*, pages 297–309. Springer Berlin / Heidelberg, 2006. 10.1007/11805588_21.

[120] N.S. Rosa, P.R.F. Cunha, and G.R.R. Justo. Processnfl: a language for describing non-functional properties. In *System Sciences, 2002. HICSS. Proceedings of the 35th Annual Hawaii International Conference on*, pages 3676 –3685, jan. 2002.

[121] James Rumbaugh, Ivar Jacobson, and Grady Booch. *The Unified Modeling Language Reference Manual*. Addison-Wesley, Boston, MA, 2. edition, 2005.

[122] Stephen M. Rutner, Brian J. Gibson, and Susan R. Williams. The impacts of the integrated logistics systems on electronic commerce and enterprise resource planning systems. *Transportation Research Part E: Logistics and Transportation Review*, 39:83–93, 2003.

[123] Ravi S. Sandhu, Edward J. Coyne, Hal L. Feinstein, and Charles E. Youman. Role-based access control models. *Computer*, 29(2):38–47, 1996.

[124] Davide Sangiorgi. Asynchronous process calculi: the first- and higher-order paradigms. *Theoretical Computer Science - Special issues on models and paradigms for concurrency*, 253(2):311–350, 2001.

[125] Amit Sheth. Semantic web process life cycle. role of semantics in annotation, discovery, composition and execution. In *Invited talk at WWW 2003 Workshop on e-Services and the Semantic Web,Budapest, Hungary.*, 2003.

[126] Halvard Skogsrud, Boualem Benatallah, and Fabio Casati. Trust-serv: model-driven lifecycle management of trust negotiation policies for web services. In Stuart I. Feldman, Mike Uretsky, Marc Najork, and Craig E. Wills, editors, *WWW*, pages 53–62. ACM, 2004.

[127] R. G. Smith. The contract net protocol: High-level communication and control in a distributed problem solver. *IEEE Transactions on Computers*, 29(12):1104–1113, 1980.

[128] Marc Spielmann. Verification of relational tranducers for electronic commerce. In *Proceedings of the nineteenth ACM SIGMOD-SIGACT-*

SIGART symposium on Principles of database systems, PODS '00, pages 92–103, New York, NY, USA, 2000. ACM.

[129] Mudhakar Srivatsa, Arun Iyengar, Thomas A. Mikalsen, Isabelle Rouvellou, and Jian Yin. An access control system for web service compositions. In *Proceedings of the IEEE International Conference on Web Services (ICWS)*, pages 1–8, 2007.

[130] Ferucio Laurentiu Tiplea and Geanina Ionela Macovei. E-timed workflow nets. In *Proceedings of the Eighth International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC)*, pages 423–429. IEEE Computer Society, 2006.

[131] Gianluca Tonti, Jeffrey M. Bradshaw, Renia Jeffers, Rebecca Montanari, Niranjan Suri, and Andrzej Uszok. Semantic web languages for policy representation and reasoning: A comparison of kaos. In *Proceedings of the 2nd International Semantic Web Conference (ISWC2003)*, pages 419–437. Springer, 2003.

[132] Antonio Vallecillo, Vasco T. Vasconcelos, and António Ravara. Typing the behavior of objects and components using session types. *Electronic Notes in Theoretical Computer Science*, 68(3):439 – 456, 2003. Foclasa 2002, Foundations of Coordination Languages and Software Architectures (Satellite Workshop of CONCUR 2002).

[133] Christopher Van, Eenoo Osama, Hylooz Khaled, and M. Khan. Addressing non-functional properties in software architecture using adl. In *In Proceedings of the 6th Australian Workshop on Software and Systems Architectures - AWSA'05, Brisbane, Australia.*, page 6–13, March 29 2005.

[134] W. M. P. van der Aalst. The application of petri nets to workflow management. *Journal of Circuits, Systems, and Computers*, 8(1):21–66, 1998.

[135] Wil van der Aalst, Lachlan Aldred, Marlon Dumas, and Arthur ter Hofstede. Design and implementation of the yawl system. In *Advanced Information Systems Engineering*, volume 3084 of *Lecture Notes in Computer Science*, pages 281–305. Springer Berlin / Heidelberg, 2004. 10.1007/978-3-540-25975-6_12.

[136] W.M.P. van der Aalst and A.H.M. ter Hofstede. YAWL: Yet Another Workflow Language. *Information Systems*, 30(4):245–275, 2005.

[137] Victor Vianu. Automatic verification of database-driven systems: a new frontier. In *Proceedings of the 12th International Conference on Database Theory*, ICDT '09, pages 1–13, New York, NY, USA, 2009. ACM.

[138] Steve Vinoski. More web services notifications. *IEEE Internet Computing*, 8(3):90–93, 2004.

[139] Steve Vinoski. Web services notifications. *IEEE Internet Computing*, 8(2):86–90, 2004.

[140] Steve Vinoski. Ws-nonexistent standards. *IEEE Internet Computing*, 8(6):94–96, 2004.

[141] Andreas Wombacher, Peter Fankhauser, Bendick Mahleko, and Erich Neuhold. Matchmaking for business processes based on choreographies. In *Proceedings of the 2004 IEEE International Conference on e-Technology, e-Commerce and e-Service (EEE)*, pages 359–368, Washington, DC, USA, 2004. IEEE Computer Society.

[142] Roosdiana Wonohoesodo and Zahir Tari. A role based access control for web services. In *Proceedings of the 2004 IEEE International Conference on Services Computing (SCC)*, pages 49–56, Washington, DC, USA, 2004. IEEE Computer Society.

[143] Daniel M. Yellin and Robert E. Strom. Protocol specifications and component adaptors. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 19(2):292–333, 1997.

[144] Liangzhao Zeng, Boualem Benatallah, Marlon Dumas, Jayant Kalagnanam, and Quan Z. Sheng. Quality driven web services composition. In *Proceedings of the 12th international conference on World Wide Web(WWW)*, pages 411–421, New York, NY, USA, 2003. ACM.