Composition de Services Web: Une Approche basée Liens Sémantiques

Freddy Lécué

Ecole Nationale Supérieure des Mines
SAINT-ETIENNE

N° d'ordre : 492 I

THESE

présentée par

Freddy Lécué

Pour obtenir le grade de Docteur
de l'Ecole Nationale Supérieure des Mines de Saint-Etienne

Spécialité : Informatique

Composition de Services Web: Une Approche basée Liens Sémantiques

Soutenue à Paris (Forum des Halles) le 8 Octobre 2008

Membres du jury

Dr. Bernd Amann - President. University Pierre and Marie Curie (LIP6), Paris, France.

Pr. Claude Godart (Hdr)- Rapporteur. Université Henri Poincaré, Nancy, France
Dr. Marco Pistore - Rapporteur. Fondazione Bruno Kessler, Trento, Italy
Dr. Biplav Srivastava - Rapporteur. IBM's T.J.Watson Research, Center in Hawthorne, NY, USA

Dr. Fabien Gandon - Examinateur. Centre de Recherche INRIA, Sophia-Antipolis, France.
Pr. Mohand-Saïd Hacid (Hdr)- Examinateur. Université Claude Bernard (LIRIS), Lyon, France

Pr. Olivier Boissier (Hdr)- Directeur de thèse. École Nationale Supérieure des Mines, Saint-Etienne, France.
Dr. Alexandre Delteil - Co- Directeur. Orange Labs, Issy, France.
Dr. Alain Léger - Co- Directeur. Orange Labs, Rennes, France.

◘ Spécialités doctorales : **Responsables :**

SCIENCES ET GENIE DES MATERIAUX	**J. DRIVER** Directeur de recherche – Centre SMS
MECANIQUE ET INGENIERIE	**A. VAUTRIN** Professeur – Centre SMS
GENIE DES PROCEDES	**G. THOMAS** Professeur – Centre SPIN
SCIENCES DE LA TERRE	**B. GUY** Maitre de recherche – Centre SPIN
SCIENCES ET GENIE DE L'ENVIRONNEMENT	**J. BOURGOIS** Professeur – Centre SITE
MATHEMATIQUES APPLIQUEES	**E. TOUBOUL** Ingénieur – Centre G2I
INFORMATIQUE	**O. BOISSIER** Professeur – Centre G2I
IMAGE, VISION, SIGNAL	**JC. PINOLI** Professeur – Centre CIS
GENIE INDUSTRIEL	**P. BURLAT** Professeur – Centre G2I
MICROELECTRONIQUE	**Ph. COLLOT** Professeur – Centre CMP

◘ Enseignants-chercheurs et chercheurs autorisés à diriger des thèses de doctorat (titulaires d'un doctorat d'Etat ou d'une HDR)

AVRIL	Stéphane	MA	Mécanique & Ingénierie	CIS
BATTON-HUBERT	Mireille	MA	Sciences & Génie de l'Environnement	SITE
BENABEN	Patrick	PR 2	Sciences & Génie des Matériaux	CMP
BERNACHE-ASSOLANT	Didier	PR 1	Génie des Procédés	CIS
BIGOT	Jean-Pierre	MR	Génie des Procédés	SPIN
BILAL	Essaïd	DR	Sciences de la Terre	SPIN
BOISSIER	Olivier	PR 2	Informatique	G2I
BOUCHER	Xavier	MA	Génie Industriel	G2I
BOUDAREL	Marie-Reine	MA	Sciences de l'inform. & com.	DF
BOURGOIS	Jacques	PR 1	Sciences & Génie de l'Environnement	SITE
BRODHAG	Christian	MR	Sciences & Génie de l'Environnement	SITE
BURLAT	Patrick	PR 2	Génie industriel	G2I
CARRARO	Laurent	PR 1	Mathématiques Appliquées	G2I
COLLOT	Philippe	PR 1	Microélectronique	CMP
COURNIL	Michel	PR 1	Génie des Procédés	SPIN
DAUZERE-PERES	Stéphane	PR 1	Génie industriel	CMP
DARRIEULAT	Michel	ICM	Sciences & Génie des Matériaux	SMS
DECHOMETS	Roland	PR 1	Sciences & Génie de l'Environnement	SITE
DESRAYAUD	Christophe	MA	Mécanique & Ingénierie	SMS
DELAFOSSE	David	PR 1	Sciences & Génie des Matériaux	SMS
DOLGUI	Alexandre	PR 1	Génie Industriel	G2I
DRAPIER	Sylvain	PR 2	Mécanique & Ingénierie	CIS
DRIVER	Julian	DR	Sciences & Génie des Matériaux	SMS
FOREST	Bernard	PR 1	Sciences & Génie des Matériaux	CIS
FORMISYN	Pascal	PR 1	Sciences & Génie de l'Environnement	SITE
FORTUNIER	Roland	PR 1	Sciences & Génie des Matériaux	CMP
FRACZKIEWICZ	Anna	MR	Sciences & Génie des Matériaux	SMS
GARCIA	Daniel	CR	Génie des Procédés	SPIN
GIRARDOT	Jean-Jacques	MR	Informatique	G2I
GOEURIOT	Dominique	MR	Sciences & Génie des Matériaux	SMS
GOEURIOT	Patrice	MR	Sciences & Génie des Matériaux	SMS
GRAILLOT	Didier	DR	Sciences & Génie de l'Environnement	SITE
GROSSEAU	Philippe	MR	Génie des Procédés	SPIN
GRUY	Frédéric	MR	Génie des Procédés	SPIN
GUILHOT	Bernard	DR	Génie des Procédés	CIS
GUY	Bernard	MR	Sciences de la Terre	SPIN
GUYONNET	René	DR	Génie des Procédés	SPIN
HERRI	Jean-Michel	PR 2	Génie des Procédés	SPIN
KLÖCKER	Helmut	MR	Sciences & Génie des Matériaux	SMS
LAFOREST	Valérie	CR	Sciences & Génie de l'Environnement	SITE
LI	Jean-Michel	EC (CCI MP)	Microélectronique	CMP
LONDICHE	Henry	MR	Sciences & Génie de l'Environnement	SITE
MOLIMARD	Jérôme	MA	Sciences & Génie des Matériaux	SMS
MONTHEILLET	Frank	DR 1 CNRS	Sciences & Génie des Matériaux	SMS
PERIER-CAMBY	Laurent	PR1	Génie des Procédés	SPIN
PIJOLAT	Christophe	PR 1	Génie des Procédés	SPIN
PIJOLAT	Michèle	PR 1	Génie des Procédés	SPIN
PINOLI	Jean-Charles	PR 1	Image, Vision, Signal	CIS
STOLARZ	Jacques	CR	Sciences & Génie des Matériaux	SMS
SZAFNICKI	Konrad	CR	Sciences de la Terre	SITE
THOMAS	Gérard	PR 1	Génie des Procédés	SPIN
VALDIVIESO	François	MA	Sciences & Génie des Matériaux	SMS
VAUTRIN	Alain	PR 1	Mécanique & Ingénierie	SMS
VIRICELLE	Jean-Paul	MR	Génie des procédés	SPIN
WOLSKI	Krzysztof	CR	Sciences & Génie des Matériaux	SMS
XIE	Xiaolan	PR 1	Génie industriel	CIS

Glossaire :

PR 1 Professeur 1ère catégorie
PR 2 Professeur 2ème catégorie
MA(MDC) Maître assistant
DR (DR1) Directeur de recherche
Ing. Ingénieur
MR(DR2) Maître de recherche
CR Chargé de recherche
EC Enseignant-chercheur
ICM Ingénieur en chef des mines

Centres :

SMS Sciences des Matériaux et des Structures
SPIN Sciences des Processus Industriels et Naturels
SITE Sciences Information et Technologies pour l'Environnement
G2I Génie Industriel et Informatique
CMP Centre de Microélectronique de Provence
CIS Centre Ingénierie et Santé

École Nationale Supérieure des Mines de Saint-Etienne, France

2005-2008

Ph.D Dissertation

A thesis submitted to the graduate faculty
in partial fulfillment of the requirements for the degree of
DOCTOR OF PHILOSOPHY
Major : Computer Science

by

Freddy Lécué

October 2008, the 8th

Web Service composition : Semantic Links based approach.

President:

Dr. Bernd Amann,	Research Staff Member at University Pierre and Marie Curie in Paris (LIP6), France

Reviewers:

Pr. Claude Godart,	Professor at the University Henri Poincaré in Nancy, France
Dr. Marco Pistore,	Research Staff Member at the Fondazione Bruno Kessler in Trento, Italy
Dr. Biplav Srivastava,	Research Staff Member at the IBM's T.J.Watson Research Center in Hawthorne, NY, USA

Examiners:

Pr. Mohand-Saïd Hacid,	Professor at University Claude Bernard in Lyon (LIRIS), France
Dr. Fabien Gandon,	Research Staff Member at the INRIA's Research Center in Sophia-Antipolis, France
Dr. Bernd Amann,	Research Staff Member at University Pierre and Marie Curie in Paris (LIP6), France

Ph.D Co-Advisor:

Pr. Olivier Boissier,	Professor at École Nationale Supérieure des Mines of Saint-Etienne, France
Dr. Alain Légér,	Scientific Direction Programme Knowledge Processing France Telecom R&D - Rennes, France
Dr. Alexandre Delteil,	Research Engineer at France Telecom R&D -Issy, France

---

Aknowledgements

I wish to thank a number of people who have supported, directed, and assisted me in completing this thesis.

If I were to name a person who deserves the most thanks, it would be my wife Sophie Lécué.

First of all she always pushed me to pursue a Ph.D. and did not hesitate to follow me in Rennes, to provide me with the necessary help when I decided to do so.

Secondly I would like to thank her for her constant love and support. I am indebted to my wife for her tireless encouragement. Her presence, at happy and hard times, and invaluable moral help were instrumental to the conduct of my research. I again thank her for all her patience and support.

I would like to thank my co-advisors Alain Léger, Olivier Boissier and Alexandre Delteil for their full support during my studies. I am very fortunate to have had the opportunity to work under their supervision.

In particular Alain instilled a thirst for excellence in me, taught me how to do scholarly research, and helped me think creatively and independently. His guidance and patience during my Ph.D. research are greatly appreciated. I will never forget my enjoyable experience working with Alain.

I would like also to express my gratitude to Prof. Claude Godart, Prof. Marco Pistore and Dr.Biplav Srivastava for serving on my Thesis committee. I also gratefully acknowledge Dr. Bernd Amann, Prof. Mohand-Said Hacid and Dr. Fabien Gandon, for kindly agreeing to be the co-examiners of my thesis and for their helpful comments about this work.

I thank my co-authors: Piergiorgio Bertoli, Philippe Bron, Raul Garcia-Castro, Jörg Hoffmann, Radu Jurca,Aurélien Moreau, Eduardo Silva, Alain Pastor, Luís Ferreira Pires, David Portabella, Samir Salibi, Maarten Steen for their productive and enjoyable collaborations.

I would like also to thank the anonymous reviewers for their comments on earlier drafts of my papers.

Much of this thesis is a direct result of a research project funded by France Telecom R&D and a PhD Grant CIFRE 159/2006 from the French National Association for Research and Technology (ANRT).

Other research projects I had the luck of taking part in are the KNOWLEDGE WEB and SPICE projects sponsored by the European Commission. I am grateful for the opportunity to work as a member of the working group on "Web service composition and the semantic Web", and I would like to thank again Marco Pistore and Mariano Belaunde for their leadership.

Last but not least, I would like also to thank Ecóle Nationale Supérieure des Mines de Saint-Etienne and more specially the Multi Agent System group members.

Contents

List of Figures

List of Tables

List of Algorithms

Abstract

Automated composition of Web services or the process of forming new value added Web services is one of the most promising challenges facing the Semantic Web today. Semantics enables Web service to describe capabilities together with their processes, hence one of the key elements for the automated composition of Web services. In this Ph.D study we focus on the functional level of Web services i.e., services are described according i) to some input, output parameters semantically enhanced by concepts in a domain ontology and ii) to preconditions and side-effects on the world. Web service composition is then viewed as a composition of semantic links controlled by causal laws. The semantic links refer to semantic matchmaking between Web service parameters (i.e., outputs and inputs) in order to model their connection and interaction whereas causal laws are the relationships between actions, action preconditions and side-effects. The key idea is that the matchmaking enables us, at run time, finding semantic compatibilities among independently defined Web service descriptions. By considering such a level of composition we first study semantic links in details, and more specially their properties of validity and robustness.

From this and depending on services expressivity we focus on two different approaches to perform Web service composition. In the first approach a formal model to perform the automated composition of Web services by means of semantic links i.e., Semantic Link Matrix is introduced. This Semantic Link Matrix is required as a starting point to apply problem-solving techniques such as regression (or progression)-based search for Web service composition. The model supports a semantic context and focuses on semantic links in order to find correct, complete, consistent and robust plans as solutions. In this part an innovative and formal model for an Artificial Intelligence planning-oriented composition is presented.

In the second approach, besides semantic links, causal laws are also considered to achieve compositions of Web services. To this end an augmented and adapted version of the logic programming language Golog i.e., s_{sl}Golog is presented as a natural formalism not only for reasoning about the latter links and laws, but also for automatically composing services. s_{sl}Golog operates as an offline interpreter that supports n-ary output parameters of actions to compute conditional compositions of services. This approach is much more restrictive since assumes more expressivity on Web service description.

Finally, since Web services have been enhanced with formal semantic descriptions, the quality of semantic links involved in a composition is used as a innovative and distinguishing criterion to estimate its overall semantic quality. Therefore non functional criteria such as quality of service (QoS) are no longer considered as the only criteria to rank compositions satisfying the same goal. In this part we focus on quality of semantic link based Web service composition. To this end, we present a general and extensible model to evaluate quality of both elementary and composition of semantic links. From this, we introduce a global semantic link selection based approach to compute the optimal composition. This problem is formulated as an optimization problem which is solved using efficient integer linear programming methods.

Our system is implemented and interacting with Web services dedicated to Telecommunication scenarios in use. The evaluation results showed high efficiency and effectiveness of the

suggested approaches.

Keywords

[*Service Computing*] Web Service, Service Composition, Composition Optimization;
[*Artificial Intelligence*] Semantic Web, Knowledge Representation, Automated Reasoning, AI Planning.

Abstract

La composition automatisée de services Web ou le processus de formation de nouveaux services Web à plus forte valeur ajoutée est l'un des plus grand défis auxquels le Web sémantique est face aujourd'hui. La sémantique permet d'un côte de décrire les capacités des services Web mais aussi leurs processus d'exécution, d'où un élément clé pour la composition automatique de services Web. Dans cette étude de doctorat, nous nous concentrons sur la description fonctionnelle des services Web c'est-à-dire, les services sont vus comme une fonction ayant des paramètres i) d'entrée, de sortie sémantiquement annotés par des concepts d'une ontologie de domaine et ii) des conditions préalables et effets conditionnels sur le monde. La composition de services Web est alors considérée comme une composition des liens sémantiques où les lois de cause à effets ont aussi un rôle prépondérant. L'idée maîtresse est que les liens sémantiques et les lois causales permettent, au moment de l'exécution, de trouver des compatibilités sémantiques, indépendamment des descriptions des services Web. En considérant un tel niveau de composition, nous étudions tout d'abord les liens sémantiques, et plus particulièrement leurs propriétés liées à la validité et la robustesse.

A partir de là et dépendant de l'expressivité des services Web, nous nous concentrons sur deux approches différentes pour effectuer la composition de services Web. Lors de la première approche, un modèle formel pour effectuer la composition automatique de services Web par le biais de liens sémantiques i.e., Matrice de liens sémantiques est introduite. Cette matrice est nécessaire comme point de départ pour appliquer des approches de recherche basées sur la régression (ou progression). Le modèle prend en charge un contexte sémantique et met l'accent sur les liens sémantiques afin de trouver des plans corrects, complets, cohérents et robustes comme solutions au problème de composition de services Web. Dans cette partie un modèle formel pour la planification et composition de services Web est présenté.

Dans la seconde approche, en plus de liens sémantiques, nous considérons les lois de causalité entre effets et pré-conditions de services Web pour obtenir les compositions valides de services Web. Pour ceci, une version étendue et adaptée du langage de programmation logique Golog (ici sslGolog) est présentée comme un formalisme naturel non seulement pour le raisonnement sur les liens sémantiques et les lois causales, mais aussi pour composer automatiquement les services Web. sslGolog fonctionne comme un interprète qui prend en charge les paramètres de sortie de services pour calculer les compositions conditionnelles de services. Cette approche (beaucoup plus restrictive) suppose plus d'expressivité sur la description de service Web.

Enfin, nous considérons la qualité des liens sémantiques impliqués dans la composition comme critère novateur et distinctif pour estimer la qualité sémantique des compositions calculées. Ainsi les critères non fonctionnels tels que la qualité de service(QoS) ne sont plus considérés comme les seuls critères permettant de classer les compositions satisfaisant le même objectif. Dans cette partie, nous nous concentrons sur la qualité des liens sémantiques appartenant à la composition de service Web. Pour ceci, nous présentons un modèle extensible permettant d'évaluer la qualité des liens sémantiques ainsi que leur composition. De ce fait, nous introduisons une approche fondée sur la sélection de liens sémantiques afin de calculer la composition optimale. Ce problème est formulé comme un problème d'optimisation qui est résolu à l'aide de la méthode par programmation linéaire entière.

Notre système est mis en oeuvre et interagit avec des services Web portant sur de scénarios de télécommunications. Les résultats de l'évaluation a montré une grande efficacité des différentes approches proposées.

Keywords

[*Calcul basé Service*] Service Web, Composition de Web services, Optimisation de compositions; [*Intelligence Artificielle*] Web Sémantique, Représentation des connaissances, Raisonnement Automatique, Planification en Intelligence Artificielle.

Introduction

General Context

Service oriented computing (SOC [158]) is an emerging cross-disciplinary paradigm for distributed computing that is changing the way software applications are designed, architected, delivered and consumed. Services are autonomous, platform-independent computational elements that can be described, published, discovered, orchestrated and programmed using standard protocols to build networks of collaborating applications distributed within and across organizational boundaries. Web Services [8] are the current most promising technology to provide the feature richness, flexibility and scalability needed by enterprises to manage the Service oriented architecture (SOA) and SOC challenges [186]. Therefore Web services provide the basis for the development and execution of business processes that are distributed over the network and available via standard interfaces and protocols.

The commonly accepted and *minimal* framework for services, referred to as SOA, consists of the following basic roles: i) the *service provider*, which is the subject (e.g. an organization) providing services; ii) the *service directory*, which is the subject providing a repository/registry of service descriptions, where providers publish their services and requestors find services; and iii) the *service requestor*, also referred to as client, which is the subject looking for and invoking the service in order to fulfil some goals. A requestor discovers a suitable service in the directory, and then connects to the specific service provider in order to invoke the service.

Research on Web services spans over many interesting issues. In this thesis, we are particularly interested in *automated composition of semantic Web services*. This issue falls into the following *Research areas*:

- *Service Computing*: Web Service, Service Composition, Composition Optimization;
- *Artificial Intelligence*: Semantic Web, Knowledge Representation, Automated Reasoning, AI (Artificial Intelligence) Planning.

Motivation and Aim

Recent progress in the field of Web services makes it practically possible to publish, locate, and invoke applications across the Web. This is a reason why more and more companies and organizations now implement their core business and outsource other application services over Internet. Thus the ability of efficient selection and integration of inter-organizational services on the Web at runtime becomes an important issue to the Web service provision. The general problem is about how to develop mechanisms to automatically locate the correct Web service

in order to meet the user's requirements. In some cases, if no single Web service can satisfy the functionality required by the user, there should be a possibility to combine existing services together in order to fulfil the request. This new ability is so called *Web service composition*:

> *Starting from an initial set of services, Web service composition aims at selecting and inter-connecting services provided by different partners according to a goal to achieve.*

The result of the Web service composition is generated in the user interaction loop on the basis of the service requests and the available services.

The problem of Web service composition is a highly complex task. Here we underline some sources of its complexities:

- First, the amount of available Web services is huge, and it is already beyond the human's capability to analyze them manually.
- Second, Web services can be created and updated on the fly, thus the composition system needs to detect the updating at runtime and the decision should be made based on the up-to-date information.
- Third, the Web services are usually developed by different organizations that use different conceptual models for presenting services' characteristics. This requires utilization of relevant semantic information for matching and composition of Web services.

The vision that we pursue in this thesis is the folowing:

> *The realization of Web service composition through the emerging of semantic Web services.*

Closing the gap between Web service composition and automatic reasoning is the aim of the emerging research area of semantic Web services. This research branch aims to identify ways of leveraging today's computational power to automate composition tasks that are currently carried out manually.

Web service composition enhanced by semantic technologies is currently one of the most hyped and addressed issue in the two major trends in Web technologies i.e., Web services and semantic Web [140].

The semantic Web i.e., the Web of meaning is considered as the new vision and extension of the current Web that tries to give semantic to the Web resources [29].

Semantic Web aims at improving the technology to organise, search, integrate, and evolve Web-accessible resources (e.g., Web documents, data) by using rich and machine-understandable abstractions for the representation of resources semantics. Ontologies are proposed as means to address semantic heterogeneity among Web-accessible information sources and services. They are used to provide meta-data for the effective manipulation of available information including discovering information sources and reasoning about their capabilities.

In the context of Web services, ontologies promise to take interoperability a step further by providing rich descriptions and modelling of service properties, capabilities, and behaviour.

Web services in the semantic Web are enhanced using rich description languages (through Description Logics DLs [58]) such as the Web Ontology Language (OWL [161]). In this way semantic Web services [197] are Web services that have been enhanced with formal semantic descriptions where OWL-S [12], the Web Service Modelling Ontology (WSMO [60]), Semantic

Annotation for WSDL (SA-WSDL [192]) or the Semantic Web Services Language (SWSL[2] [179]) may be used to describe them.

Services requestor (e.g., other services) can, in turn, use these descriptions to reason about Web services and automate their use to accomplish goals specified by end-users including intelligent and automated discovery, selection, composition and invocation [197].

The aim of this thesis is to consider the Web service composition problem from the semantic viewpoint, and to propose a framework for supporting the composition process. The semantic based method ensures the correctness and the completeness of the solution.

Research Questions and Contributions

In this report, we address the following problem:

> Automated composition of stateless Web services in the semantic Web and semantic-based optimization of candidate compositions.

Towards this issue we defend the following thesis:

> Causal laws between services and semantic links between their input/output parameters are key elements for automated composition of stateless Web Services and for the optimisation of their candidate compositions.

A major contribution of this thesis is the development and specification of a formal and generic approach for the automatic composition of Web services on the Semantic Web. In order to accomplish such advances in the field of *semantic Web services* and more specially of their *composition*, many open problems have to be solved both on the theoretical and on the practical level.

In this thesis we aim to treat nine questions i.e., **seven research questions** and **two questions related to the implementation, validation and demonstration** of our model in **industrial scenarios in use**. On the one hand the seven research questions are structured in the following four main classes:

- **Semantic Composability**[3] **of Web Services** or *how to ensure composition of Web services?*
- **Web Service Composition** or *how to combine Web services in a suitable way to achieve a target goal?*
- **Composition Optimization** or *how to select an optimal composition among a set of candidates that achieve the same goal?*
- **Architecture** or *how to perform an end-to-end composition of Web services?*

On the other hand the two last questions are denoted by the following class:

[2] http://www.daml.org/services/swsl/

[3] The reader can interchange the *semantic composability* term with the *data flow* term without major modification of the meaning.

- **Industrial Scenarios and Implementation** or *how implementing, validating, and demonstrating the composition approach in industrial scenarios in use?*

In the following we address these different classes with the next questions.

Semantic Composability of Web Services

Q1. How can the semantic composability of Web Services be ensured?

- This thesis' contribution to answering this question is *a novel semantic link concept* [114, 115, 113, 119, 112] between Web services which is a composability criterion considered as the main issue to form new value-added services by composition at functional level. This criterion is required to semantically link output to input parameters of Web services, and to ensure correct properties of semantic composability involved between Web services.

Q2. What kind of properties the semantic composability of Web services have to satisfy?

- This thesis' contribution to answering this question is *robustness of its semantic links* [108, 109, 112]. The thesis highlights the issue related to robustness of semantic links and presents different techniques to solve the problem of robustness in Web service composition.

Q3. How to define a formal and flexible model that ensures correct semantic composability of Web services and easily supports basic control flow of composition?

- This thesis' contribution to answering this question is *a framework SLM (Semantic Link Matrix)* [114, 115, 113, 119, 112] for pre-computing and capturing relevant semantic links in a formal and flexible model to achieve Web service composition. Moreover this model supports expressive compositions of Web services such as sequence, non determinism choice of service and concurrent execution.

Web Service Composition

Q4. How can we compose Web services with semantic links?

- This thesis' contribution to answering this question is Ra_4C (**R**egression-based **A**pproach **for C**omposition) [114, 115, 112] that applies an AI planning-based technique for computing correct, consistent, complete and optionally robust Web service compositions on an SLM of a given domain. The result of the latter computing is a partial ordering of Web services arranged in a simpler version of a workflow that fulfils a given composition goal. The thesis fits semantic Web service composition to a semantic links composition by means of the feature of sequence composability between services.

Q5. How can we compose semantic Web services wherein conditions on their functional parameters hold?

- This thesis' contribution to answering this question is *a composition approach that considers semantic links together with causal laws as composability criteria.* This approach, which is based on an adaptation and extension of Golog i.e., s_{sl}Golog (Sensing + Semantic Link for Golog), supports conditional compositions of services.

Composition Optimization

Q6. How can we compare and select among a huge number of composition solutions that achieve the same goal?

- This thesis' contribution to answering this question is *a technique for computing the semantic link based optimal Web service composition* [110]. To this end, a general and extensible model to evaluate and estimate quality of both elementary and semantic links based Web service composition is presented. From this the semantic link based *optimal* Web service composition is computed by i) a local, ii) a naive and iii) a global selection approach.

Architecture

Q7. How can we achieve an end to end composition of Web services at functional level?

- This thesis' contribution to answering this question is *a reference architecture* [168, 30, 112, 71, 70] *wherein Web services can be discovered, selected, composed and executed at Functional Level*. All these components interact with repository of Web services, semantic reasoners to achieve such a goal.

Industrial Scenarios and Implementation

The two questions related to our industrial context are as follows:

Q8. Are the suggested approaches implemented?

- Our Ph.D work has an efficient implementation in the SME^3-Pro[4] prototype for automatically computing compositions of Web services and rendering them in a standard format.

Q9. Are the suggested approaches running on real scenarios and scaling large use cases?

- The thesis illustrates our approaches in different levels of scenarios i.e., from scenarios in Use in Telecom, E-Tourism or E-HealthCare domain, to random scenarios to evaluate scalability of our approach. Moreover the empirical evaluation illustrates the experiences made with the application of the proposed techniques and leads to the conclusion that the concepts presented can be applied to practical problems.

Thesis Outline

Figure 1 illustrates the organization and inter dependences of the different Ph.D report's Chapters. In more details, this thesis is organized as follows:

- Part I presents the background information to follow the theory in this thesis. Moreover this part presents related works, that we first analyse with the previous questions $Q_{i,1\leq i\leq 9}$. Then we emphasize the main limitations of the latter works by means of the latter questions. From this, the part draws main requirements that we consider to achieve Web service composition.

[4] cf. the SME^3-Pro (**S**e**M**antic w**E**b s**E**rvic**E** **PRO**ject) Open Source Project licensed under the GPL license available at https://sws-orangelabs.elibel.tm.fr/ that is the general framework in which we intend to release the various prototypes produced by our research in area of Semantic Web services.

- **Chapter** 1 provides an introduction to the main topics and background information of the thesis i.e., Service-oriented Computing, Web Service, semantic Web and semantic Web service.
  This Chapter represents the foundations of Figure 1.

- **Chapter** 2 surveys a representative set of existing literature that is related to the work presented in this thesis. Moreover this Chapter poses six specific technical meta requirements that need to be met by *matchmaking* and *AI planning* based systems to make automatic web service composition at functional level a real success.
  This explains the vertical position of Chapter 2 in Figure 1.

- **Part** II presents the main contributions of this thesis.

  - In **Chapter** 3, semantic links between parameters of Web services are defined as the main composabilty criteria to achieve Web service composition. Towards this issue we focus on functional input and output parameters of services. More particularly we will consider both standard and non standard matchmaking functions to value semantic links between parameters of Web services. From this innovative valuation, new issues related to robustness connections have been identified.
    This Chapter addresses questions Q_1 and Q_2.

    Moreover **Chapter** 3 studies a formal and flexible model for Web service composition. This model is introduced for facilitating computation (at Design Time) of semantic links between services and also their composition. This model aims at supporting expressive (sequence, non determinism and concurrency) compositions of Web services.
    This Chapter addresses question Q_3. This Chapter represents the first contribution in Figure 1 and is based on Chapter 2.

  - **Chapter** 4 deals with the problem of service composition i.e., the process of selecting a set of web services that, combined in a suitable way, are able to perform a composition goal. Two complementary Web service composition approaches, depending on the level of Web service description and the composability criteria, are addressed in this Chapter. AI principles and matchmaking based systems are coupled to solve such a problem. In more details these two approaches are as follows: the first approach considers semantic links as a composability criterion, and the other approach works with semantic links together with causal laws. Both approaches support expressive compositions of Web services at different levels.
    This Chapter addresses questions Q_4 and Q_5. This Chapter is based on Chapters 2 and 3 in Figure 1.

  - Contrary to approaches that compute optimal composition by considering only non functional parameters (e.g., quality of service), **Chapter** 5 focuses on functional parameters of Web services to value *semantics* of compositions. In particular this Chapter aims at computing semantic links based optimal Web service composition. Starting from an initial set of web services, the goal of this chapter aims at selecting web services and maximizing the overall quality of their inter-connections by means of their

semantic links according to a goal to achieve. The composition results of Chapter 4 can be inputs of the optimization process.

This Chapter addresses question Q_6. This Chapter is based on is Chapters 2, 3 and 4 in Figure 1.

- **Part** III presents the prototype implementation of the theoretical Part II, running on scenarios in use at France Telecom R&D.

  - **Chapter** 6 presents three different scenarios wherein our approach of semantic Web service composition has been integrated. These three scenarios respectively refer to a Telecommunication, an E-Tourism and E-HealthCare scenarios.

    This Chapter addresses question Q_9. This Chapter provides scenarios in use for Chapters 3, 4 and 5 in Figure 1.

  - **Chapter** 7 presents the implementation of our service composition system. We also present the empirical evaluation of the system performance.

    This Chapter addresses questions Q_7, Q_8 and Q_9. This Chapter provides implementation for Chapters 3, 4 and 5 in Figure 1. Moreover the implementation is performed on Scenarios in Use of Chapter 6.

- Finally we summarize the thesis, discuss its contributions and provide a discussion of the method and future directions for this work.

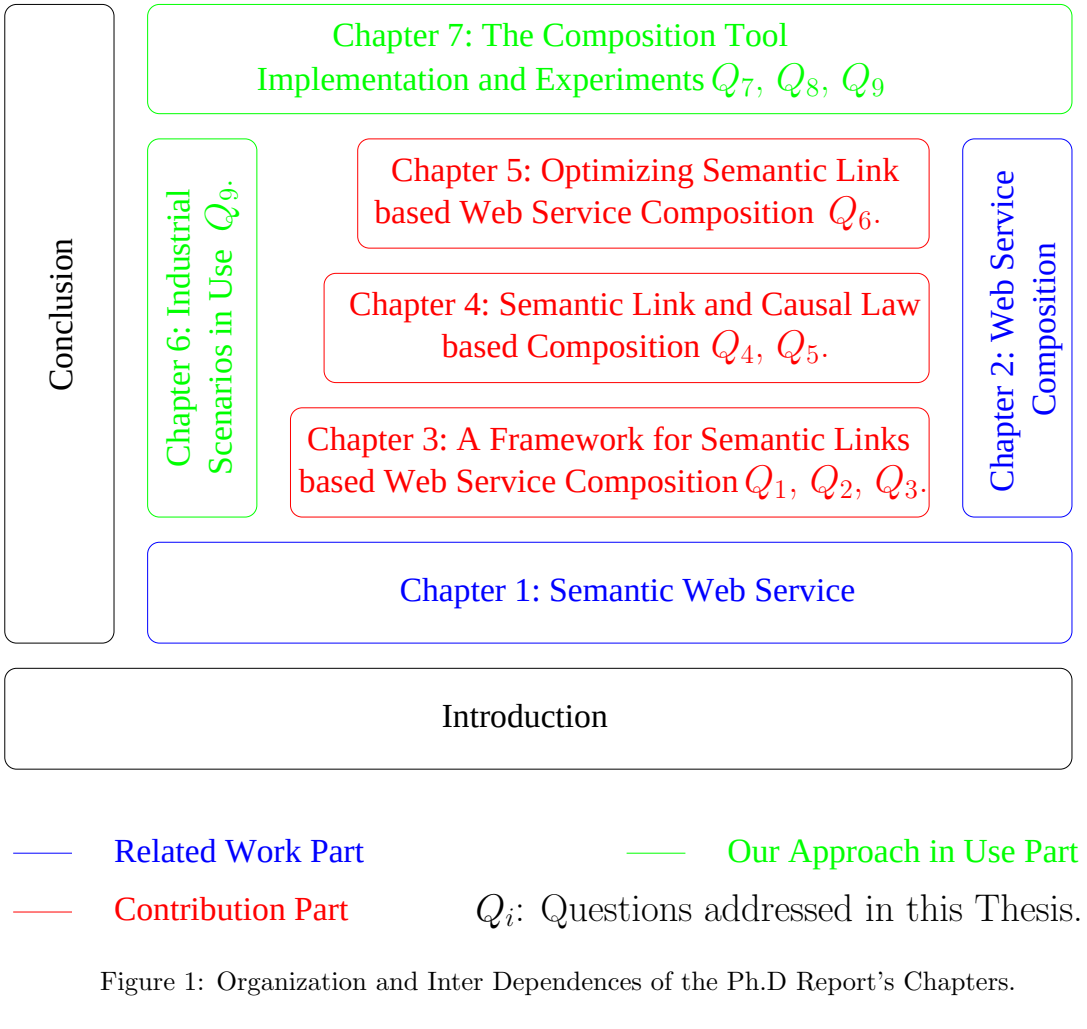


Figure 1: Organization and Inter Dependences of the Ph.D Report's Chapters.

Part I

Related Work

Chapter 1

Semantic Web Service

This chapter provides an introduction to the main topics and background of the thesis. To this end Section 1.1 introduces concepts related to service-oriented computing, service-oriented architecture and web service. In Section 1.2 we briefly review semantic web and its underlying formal model i.e., Description Logics. Section 1.3 presents semantic web services and its standard proposals. Finally Section 1.4 concludes this chapter.

1.1 Service

In this section, we introduce *Service-Oriented Computing*, its *Architecture* and *Web Service*. In the first section we will define *Service-Oriented Computing* as an emerging computing paradigm utilizing services to support the rapid development of distributed applications in heterogeneous environments. From this definition we will focus on a key architecture to realize the *Service-Oriented Computing* concept i.e., the *Service-Oriented Architecture* in the second section. The latter architecture is not tied to a specific implementation technology. Indeed it may be implemented using a wide-range of technologies. In the third section we will focus on one of these technologies i.e., *Web Services*-based SOA.

1.1.1 Service-Oriented Computing

In today's world, the ability to quickly deliver new applications is increasingly becoming imperative for business organizations. They face rapidly changing market conditions, pressure from competition and new regulations that demand compliance. These evolving constraints drive the need for the IT infrastructure to respond aptly in support of new business models and requirements. However, most of the enterprise and legacy applications were not designed to enable rapid adoption and adaptation of functionality. Therefore these become a bottleneck in the already intricate IT landscape of an organization for efficient and effective application development.

Service-Orientation aims to provide the underlying machinery that can potentially i) overcome these drawbacks and ii) realize such an "on-demand" IT environment by essentially supporting three important requirements [123]: *Integration*, *Virtualization* and *Management*. *Integration* in this context refers to the ability to seamlessly combine multiple existing, and often heterogeneous, applications and resources across organizations. *Virtualization* is the ability to provide a uniform and consolidated access to the applications irrespective of programming language used for its implementation, server hosting the application, operating system on which it is running,

and so on. And finally, *Management* is the ability to provide a logical architecture for managing computing resources using managed objects and their relationships.

The previous requirements are enabled by adopting a programming model called *Service-Oriented Computing* (i.e., SOC) [156, 158, 160]. This paradigm utilizes services as the building blocks for fast, low-cost and efficient (distributed) application development. Services are autonomous, self-describing and platform-agnostic computational entities that can perform various functions ranging from responding to simple requests to complex enterprise processes.

They allow organizations to expose multiple applications, using standard interface description languages. In addition they can be accessed and invoked programmatically over the network (Internet or intranet) using widely adopted protocols and languages. Furthermore, SOC enables the services to be published in repositories and dynamically discovered and assembled for building massively distributed, interoperable and evolvable systems. Typically, they are built in a way without preconceiving the context in which they will be used. Consequently, the provider and consumer of a particular service are loosely coupled, and often inter-organizational.

Moreover, even though there is an abundance of Web-based applications primarily targeted for humans, services are also meant to be used by other applications (and possibly by other services) directly, and not only by humans. In other words, the goal of SOC is to enable pure service-to-service interactions as opposed to only service-to-human interactions.

A key architecture to realize the SOC concept is the *Service-Oriented Architecture* (i.e., SOA). In the same direction as SOC, SOA introduces a new philosophy for building distributed applications where elementary services can be published, discovered and bound together to create more complex valued-added services. In the next section, a definition of SOA together with its main components are remained to the reader.

1.1.2 Service-Oriented Architecture

One of the most widely adopted ways for realizing the SOC model into an architecture is called a *Service-Oriented Architecture* [56, 61]. It is in essence a logical way for developing distributed software system by providing services to end-user applications or to other services via published and discoverable interfaces. OASIS[1] (i.e., the Organization for the Advancement of Structured Information Standards) defines SOA as follows:

Definition 1. ***(Service-Oriented Architecture [148])***
Service-Oriented Architecture *(SOA) is paradigm for organizing and utilizing distributed capabilities that may be under the control of different ownership domains. It provides a uniform means to offer, discover, interact with and use capabilities to produce desired effects consistent with measurable preconditions and expectations.*

In general, an SOA is composed of three different entities (Figure 1.1):

i) the service provider is any organisation that provides the service, such an entity provides a specific implementation of the service;

ii) the service requestor (e.g., a client) is an entity which searches for and invokes a particular service in order to fulfil its goals;

iii) and finally a discovery engine which acts as a repository or a directory of services.

[1] http://www.oasis-open.org/

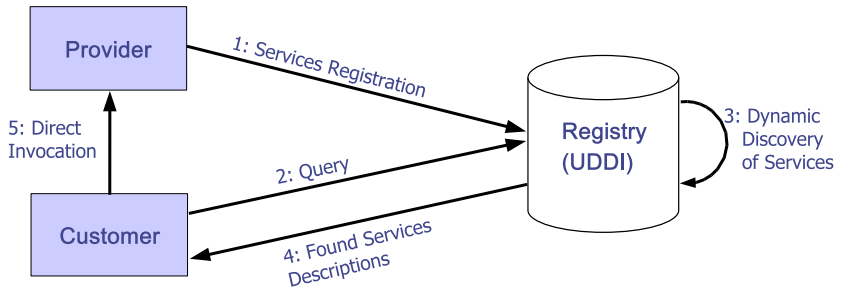


Figure 1.1: Diagram of a *Service-Oriented Architecture*.

In such an architecture the different entities act as follows. First a provider publishes information about the interface description of the service, and about the invocation (e.g., URL, protocols) in a discovery engine. Then, a client who wants to use a particular service, searches for it in the repository and then interacts with the service provider (directly or not e.g., through a broker, a delegator) for service invocation.

Even though this framework is simplistic, it raises very interesting research issues with respect to:

- *specification* e.g., how to specify the syntax and semantics of a service unambiguously?
- *publication* e.g., how to make information about a Web service available?
- *discovery* e.g., how to find the best service suitable for a particular job?
- **composition e.g., how to assemble multiple atomic and composite services for a particular job?**
- *selection* e.g., how to select "*best*" services for a given task?
- *interoperation* e.g., how to solve the problem of heterogeneity at a data level among Web services?
- *mediation* e.g., how to provide direct *connectivity* between service requestors and service providers?
- *interaction* e.g., what kind of data and protocol mediators are required by service requestors and service providers?
- *verification* e.g., how to validate Web services, their compositions ?
- *monitoring/management* e.g., how to provide quantified QoS-based services, service assurance to Network Providers? How to manage network resources?
- *execution* e.g., how to execute services securely?

As mentioned earlier, the main focus of our work is to develop techniques for automatic composition of services defined over the Web.

To conclude, it is obvious that a well-constructed, standards-based *Service Oriented Architecture* can empower a business environment with a flexible infrastructure and processing environment. SOA achieves this by provisioning independent, reusable automated business process and systems functions as services and providing a robust and secure foundation for leveraging these services. Efficiencies in the design, implementation, and operation of SOA-based systems can allow organizations to adapt far more readily to a changing environment.

SOA is not tied to a specific implementation technology. Indeed it may be implemented using a wide-range of technologies including *Web Services* [8], RPC [33], DCOM [67], REST [63] or CORBA [149]. In our work, we focus on *Web Services*-based SOA [157], which is described in the following section.

1.1.3 Web Service

Definitions

Web Services are the current most promising technology based on the concept of SOC and SOA [203]. According to IBM[2], "*Web Services are self-contained, modular applications, accessible via the Web through open standard languages, which provide a set of functionalities to businesses or individuals*".

This definition places the emphasis on two points. The first point is that a Web service is seen as an application accessible to other applications over the Web. Secondly, Web services are open, which means that services have published interfaces that can be invoked by message passing standards.

This definition is very simple, but not precise enough. For instance, it is not clear what it is meant by a modular, self-contained application.

A step further in refining the definition of Web service is the one provided by the World Wide Web consortium (i.e., W3C):

Definition 2. ***(Web Service)***
A Web service is a software system identified by a URI and designed to support interoperable machine-to-machine interaction over a network. It has an interface defined and described in a machine-processable format (specifically WSDL [46]). Its definition can be discovered by other software systems. Other systems may then interact with the Web service in a manner prescribed by its description using SOAP [81] messages, typically conveyed using HTTP with an XML [37] serialization in conjunction with other Web-related standards.

The W3C definition stresses that Web services should be capable of being "defined, described, and discovered," thereby clarifying how to access the Web services [8] with *simple* descriptions. We should also emphasize that Web services do not merely provide *static* information, but allow one to affect some action or change in the world (preconditions, effects), e.g., the sale of a product, the control of a physical device, and so on.

This definition, represented pictorially in Figure 1.2, outlines two fundamental requirements of Web services:

[2]IBM Web services tutorial. Online: http://www-106.ibm.com/developerworks/webservices/.

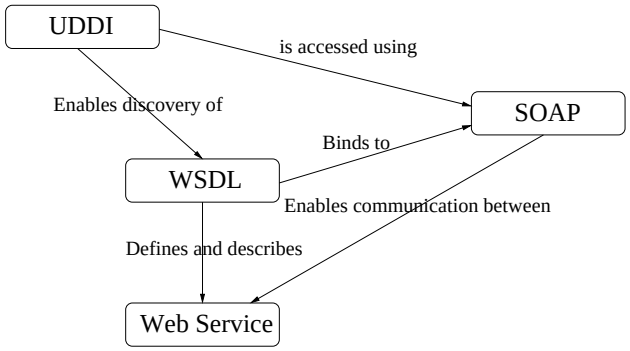


Figure 1.2: Relationships Between Standard Web Service Specifications [56].

- they communicate by exchanging data formatted as XML documents using SOAP over Internet protocols (such as HTTP);
- they provide a service description that, at minimum, consists of a WSDL document (which unfortunately does not describe Web service semantics);

where, SOAP provides a standard, extensible, and composable framework for packaging and exchanging XML messages and WSDL describes Web services starting with the messages that are exchanged between the service requester and provider. Thus, such a description describes a service in terms of its *syntactical* functionalities that it exports which can be invoked by *syntactic* input/output messages.

UDDI (for Universal Description Discovery and Integration), an open industry initiative, sponsored by OASIS, was originally proposed as a core Web service standard. It is designed to be interrogated by SOAP messages and to provide access to Web Services Description Language documents describing the protocol bindings and message formats required to interact with the web services listed in its directory. A UDDI business registration consists of three components:

- *White Pages*: address, contact, and known identifiers;
- *Yellow Pages*: industrial categorizations based on standard taxonomies;
- *Green Pages*: technical information about services exposed by the business.

Functional and Process Level Based Web Services: Two Types of Description

In practice, there are two basic types of Web services: *functional* level based Web services and *process* level based Web services. The two previous types of Web services are respectively described at *Functional level* and at *Process level*.

On the one hand *functional level based* services are single network-accessible applications that can be invoked by sending a message. Upon invocation, the service performs its task and (in some cases) produces a response to the invoker. Thus, there is no ongoing interaction between the service requestor and the service itself.

The description of such Web services focuses on their *functionality* in terms of *service name*, *operations names*, *message names* (also known as input and output messages/parameters), *interface name*.

Examples of services that would fall into this category include services which given the zip code of a city will output the current temperature or given the symbol of a company will provide the current stock quote.

The *process level based* Web services, on the other hand, comprise multiple operations that follow a given overall behaviour. Such services will require an extended interaction between the service requestor and the set of operations providing a particular functionality.

That is, web services are usually composite i.e., the interaction with them does not only consist of a single request-response step, but they require to follow a complex protocol in order to achieve the required result. The steps defining the complex interactions are not necessarily restricted to a simple sequence but can also consist of arbitrary (conditional and iterative) combinations of atomic interactions with conditional outcomes. In such a level of description objects manipulated by web services are typed messages and operations with complex descriptions. To this end a detailed description of each (relevant part of the) internal behaviour is required e.g., state transition systems (STS) [129, 169, 170], Finite state automata [26, 74], Mealy machine [95] or Interface automata [68].

In the same way as the *functional level based* services, the *process level* description focuses on their *functionality* in terms of *service name*, *operations names*, *interface name*, but also the overall *behaviour* i.e., a protocol to interact with operations described in the functional description.

Many e-Commerce sites such as Amazon.com, eBay.com fall into this category. For example, in order to purchase a digital camera at eBay, a user has to first search for it using different criteria, possibly read the reviews and analyze user's ratings, search for relevant accessories, and then finally provide payment and shipping information to complete the purchase. Such a service follows a protocol defined by the service provider.

It is obvious that *process level based* Web services have also functional description of their operations and services. However operations of *functional level based* services have no behavioural interactions. This is the main difference between these two levels. In such a case the two levels can be used as complementary levels to describe more *fine-grained* Web service.

According to this difference, processes related to, for instance, their discovery, composition are also different, and then have different issues.

What makes Web Services Attractive?

One of its most appreciated feature is the ability to integrate the Web services developed by different organizations together to fulfil the user's requirement. Such integration is based i) on the common standards of Web service interfaces, regardless of the languages that are used to implement the Web services, and ii) on the platforms where the Web services are executed.

In general, the Web services have the following features that make them better in integration inside the heterogeneous environments:

- **loosely coupling**: In software development, coupling typically refers to the degree to which software components/modules depend upon each other. Comparing with the tightly coupled components (such as the Distributed Component Object Model (DCOM) [142] or the Common Object Request Broker Architecture (CORBA) [149]), the Web services are

autonomous and can operate independently from one another. The loosely coupled feature enables Web services to locate and communicate with each other dynamically at runtime.

- **universal accessibility**: The Web services can be defined, described and discovered through the Web that enables an easy accessibility. Not only the Web services users can locate appropriate services, but services can describe and advertise themselves so that they are possible to bind and interact with each other.

- **standard languages**: Web services are described by standard XML languages that have been considered as parts of the Web technology. The Web services standards are of higher abstraction. Although the cores of Web services may be implemented by different programming languages, the interface of Web services are described by uniform standard XML languages.

The standard may be of the most importance for the success of Web services. In fact, the Web service community has proposed dozens of standard languages and frameworks to help users to present the services in a uniformed matter. In this direction the Web service languages proposed or co-proposed by IBM are considered as the most elaborated and the best described industry quasi-standard for Web service so far. The IBM Web service languages represent a traditional view of Web service languages.

Some Limitations

According to the previous view, Web services have *simple* rather than *rich* descriptions and data exchange are *syntactic* rather than *semantic* [194].

For Web services to interact properly with each other as part of composite applications that perform more complex functions by orchestrating numerous services and pieces of information, the requester and provider entities must agree not only on the service description (i.e., the WSDL definition) but also on semantics that will govern the interaction between them. Such semantic consideration seems key for the future of Web services.

Towards this issue, a complete semantic solution requires that semantics are addressed not only at the terminology level but also at the level that Web services are used and applied in the context of business scenarios, viz. at the business process-level. This implies that there must be agreement between a service requester and provider as to the implied processing of messages exchanged between interacting services that are part of a business process.

It is then widely acknowledged that to enable service-to-service interactions there should be a provision for services to automatically find, select and communicate with other services, which in turn requires the services to explicitly specify their "semantics" unambiguously. The semantics of a service should capture various aspects including, for instance, functional properties, behavioural (control/data-flow) properties, transactional properties, quality of service properties.

Since UDDI-based discovery is mainly based on a syntactic search (i.e., through *White, Yellow and Green* pages), it seems important to consider more sophisticated process to retrieve relevant services according to the semantic they can express. This could be achieves through a semantically-enhanced Web service registry to enhance the discovery facilities (through fine-grained matchmaking) that a typical UDDI registry can offer.

In the semantic Web where annotations of Web services are possible, a fourth player should be required in the SOA triangle (Figure 1.1) to details origins and descriptions of schemas/ontologies used by Web services. Towards this issure we can easily imagine different adaptations of Figure 1.1 e.g., ontology provided and maintained by a third entity.

In the next section we overview the semantic Web and its underlying formal model (i.e., Description Logics) which can be used to enhance the semantics of the nowadays syntactic Web services. This provides some additional and relevant modifications to move the traditional languages closer to the vision of an *openness* and *interoperability*.

1.2 Semantic Web and Description Logics

In this section we focus first on the semantic Web, then introduce its underlying formal model. In the third section we present some inference models to reason on Description Logics, hence on the semantic Web. Finally we draw some conclusions in the forth section.

1.2.1 Semantic Web

The *Semantic Web* [29] is an extension of the current Web in which information is given well-defined meaning, better enabling computers and people to work in cooperation. This is realized by marking up Web content, its properties, and its relations, in a reasonably expressive markup language with a well-defined semantics.

In such a context, some languages also known as semantic web languages are used to represent information about resources on the Web. This information is not limited to be about Web resources but can be about anything that can be identified. Uniform Resource Identifiers (URIs) are used to uniquely identify entities. For example, it is possible to assign a URI to a person, to the company she works for, to the car she owns. Therefore relations between these entities can be written and shared on the semantic Web. The stack of languages has been published as W3C recommendations to be used on *Semantic Web*. Let's describe it in the following paragraphs.

RDF, RDF-S

At the bottom layer of the stack, there is the Resource Description Framework (RDF) [38]. RDF is a simple assertional language that is designed to represent information in the form of triples. Triples are statements that contain a subject, a predicate and an object.

RDF Schema (RDF-S) [39] is a collection of RDF resources that can be used to describe properties of other RDF resources. Unlike its name suggests, RDF-S is not a schema that imposes specific constraints on the structure of a document, but instead it provides information about the interpretation of the statements given in an RDF data model. In this regard, RDF-S has similarities to frame based languages.

OWL

The Web Ontology Language (OWL) (formerly DAML) [161], is the most expressive standardized *Semantic Web* language that is layered on top of RDF and RDF-S.

OWL can be used to define classes (unary relations) and properties (binary relations) as in RDF-S but also provides constructs to create new class descriptions as logical combinations (intersections, unions, or complements) of other classes, define cardinality restrictions on properties

and so on. OWL has three different species: OWL-Lite, OWL-DL and OWL-Full. OWL-Lite and OWL-DL differ from OWL-Full such that they define certain constraints on RDF and RDF-S so as to be compatible with the traditional semantics of Description Logics.

It is acknowledged that DLs have heavily influenced the development of the semantic web languages. For example, RDF-S can even be described as a relatively inexpressive Description Logic [187] while OWL is in fact an alternative syntax for a very expressive Description Logic [93]. In the next section we present this family of formal logic.

1.2.2 Description Logics: An Overview

The semantic Web uses Description Logics (DLs) [58] as a formal framework. In this section, we first give the basic definitions of Description Logics (henceforth DL) [58] by briefly describing the syntax and semantics of the DL. Then we briefly describe the two main levels of knowledge we can model by means of DLs.

Basic Definitions

The family of Description Logics (DLs) is a knowledge representation system evolved from early frame systems [143] and semantic networks [171]. DLs are distinguished from their ancestors by having a precise semantics which enables the description and justification of automated deduction processes. In this direction DLs allow to represent domain of interest in terms of concepts or descriptions (unary predicates) that characterize subsets of the objects (individuals) in the domain, and roles (binary predicates) over such a domain. Concepts are denoted by expressions formed by means of special constructors. Let N_C be the set of concept names and N_R be the set of roles. Elementary concept descriptions are called atomic concepts. These are defined only by the word that is their concept name. Let N_A be the set of atomic concepts (thus $N_A \subseteq N_C$). Concept descriptions are inductively built from atomic concepts, roles and constructors. Examples of DL constructors considered in this Ph.D report follow:

- the symbol $\top$ is a concept description which denotes the top concept while the symbol $\bot$ stands for the bottom concept;
- the negation restricted to atomic concepts ($\neg A$), e.g., the description $\neg FastNetworkConnection$ denotes the class of network connections which are not fast (i.e., $SlowNetworkConnection$);
- concept conjunction ($\sqcap$), e.g., the concept description $NetworkConnection \sqcap Private$ denotes the class of network connections in the private domain (i.e., private network connection);
- the universal restriction ($\forall R.C$), e.g., the description $\forall mBytes.1M$ denotes the set of network connection which the bit rates are all $1M$ i.e, one megabytes by second;
- the cardinality restriction constructs ($\geq nR$) and ($\leq nR$), e.g., the description ($\geq 1PhoneNum$) denotes the class of individuals having at least one phone number, while the description ($\leq 1Email$) denotes the class of individuals that cannot have more than one email address;
- the qualified existential restriction ($\exists R.C$), e.g., the description $\exists mBytes.Max$ denotes the set of network connection which at least one of their bit rates is Max i.e., bigger than one megabytes by second;

[3] A is an atomic concept, C and D are concept descriptions, R is a role and n a positive integer or null. $\#S$ refers to the cardinal of the set S.

Constructors name	Syntax	Semantics	$\mathcal{AL}$	$\mathcal{ALE}$	$\mathcal{ALN}$
atomic concept	A	$A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$	✓	✓	✓
universal concept	$\top$	$\Delta^{\mathcal{I}}$	✓	✓	✓
bottom concept	$\bot$	$\emptyset$	✓	✓	✓
atomic negation	$\neg A$	$\Delta^{\mathcal{I}} \backslash A^{\mathcal{I}}$	✓	✓	✓
intersection	$D \sqcap C$	$D^{\mathcal{I}} \cap C^{\mathcal{I}}$	✓	✓	✓
limited existential restriction	$\exists R$	$\{x \in \Delta^{\mathcal{I}} \mid \exists y : (x,y) \in R^{\mathcal{I}}\}$		✓	
universal restriction	$\forall R.C$	$\{x \in \Delta^{\mathcal{I}} \mid \forall y : (x,y) \in R^{\mathcal{I}} \rightarrow y \in C^{\mathcal{I}}\}$	✓	✓	✓
at least cardinality restriction	$\geq\ nR$	$\{x \in \Delta^{\mathcal{I}} \mid \#\{y \mid (x,y) \in R^{\mathcal{I}}\} \geq n\}$			✓
at most cardinality restriction	$\leq\ nR$	$\{x \in \Delta^{\mathcal{I}} \mid \#\{y \mid (x,y) \in R^{\mathcal{I}}\} \leq n\}$			✓
exact cardinality restriction	$=\ n.R$	$\{x \in \Delta^{\mathcal{I}} \mid \#\{y \mid (x,y) \in R^{I}\} = n\}$			✓
qualified existential restriction	$\exists R.C$	$\{x \in \Delta^{\mathcal{I}} \mid \exists y : (x,y) \in R^{\mathcal{I}} \vee y \in C^{\mathcal{I}}\}$		✓	

Table 1.1: DL syntax and semantics of some concept-forming constructs[3] with their presence in $\mathcal{AL}$, $\mathcal{ALE}$ and $\mathcal{ALN}$.

The first two columns of Table 1.1 illustrates the previous constructors as well as their syntaxes. The third column illustrates their semantics. Note that the constructor $\exists R$ is equivalent to $\geq\ nR$ and $=\ n.R$ is equivalent to $(\geq\ nR) \sqcap (\leq\ nR)$.

The various description logics differ from one to another based on the set of constructors they allow. Table 1.1 shows which constructors are present in two $\mathcal{AL}$ ($\mathcal{A}$ttributive $\mathcal{L}$anguage) DLs:

- $\mathcal{ALE}$ ($\mathcal{A}$ttributive $\mathcal{L}$anguage and full $\mathcal{E}$xistential qualification);
- $\mathcal{ALN}$ ($\mathcal{A}$ttributive $\mathcal{L}$anguage and cardinality $\mathcal{N}$umber restrictions).

A concept built using the constructors of a description logic $\mathcal{L}$ is called an $\mathcal{L}$-concept, its associated description is called an $\mathcal{L}$-description. For instance, we can model $\mathcal{ALE}$-description of concept describing network connection with a speed of one megabytes by second (i.e., a concept illustrated in Figure 1.4)[4] as follows:

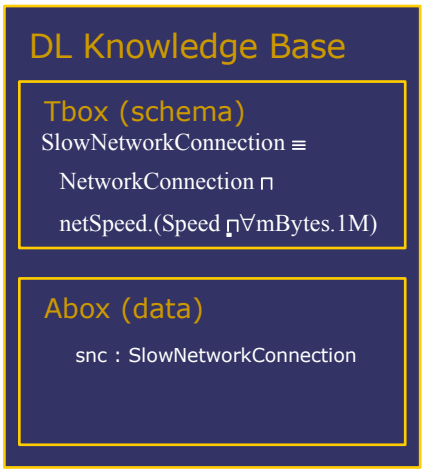
$$NetworkConnection \sqcap \forall netSpeed.(Speed \sqcap \forall mBytes.1M)$$

The semantics of a concept description (third column of Table 1.1) is defined in terms of an interpretation $I = (\Delta^{\mathcal{I}}, \Delta^{\mathcal{I}})$, which consists of a nonempty set $\Delta^{\mathcal{I}}$, the domain of the interpretation, and an interpretation function $\Delta^{\mathcal{I}}$, which associates to each concept name $P \in N_C$ a subset $P^{\mathcal{I}}$ of $\Delta^{\mathcal{I}}$ and to each role name $R \in N_R$ a binary relation $R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$. Additionally, the extension of $\Delta^{\mathcal{I}}$ to arbitrary concept descriptions is defined inductively as shown in the third column of Table 1.1.

[4]The Terminology described in Figure 1.4 represents a sample of the domain ontology of the Telecommunication scenario presented in Chapter 6.

Terminological and Assertional Knowledge

Given a DL knowledge base two main components can be distinguished: the terminological and assertional knowledge. The terminological knowledge is also denominated as the Terminological Box (henceforth TBox). The Assertional Knowledge is also denominated as the Assertional Box (henceforth ABox). In the following we describe these two levels of knowledge we can model in a DL knowledge base:



- The Terminological Box, noted $\mathcal{T}$, contains intentional (terminological) knowledge in the form of a terminology. It defines the structure of the knowledge domain and consists of a set of asserted axioms, i.e., the definition of new concepts in terms of other previously defined concepts.
- The Assertional Box contains extensional (assertional) knowledge. Such a knowledge contains a concrete knowledge domain and asserted axioms about individuals of the discourse domain, e.g., an individual is an instance of a concept or an individual is related to another by a role.

Figure 1.3: Sample of a TBox and ABox.

In other words, the TBox contains the definitions of concepts and roles, while the ABox contains the definitions of individuals (instances). Intentional knowledge is usually thought to change rarely and extensional knowledge is usually thought to be contingent, or dependent on a single set of circumstances, and therefore subject to occasional or even constant change. Figure 1.3 illustrates these two main components while Figures 1.4 and 1.5 denote parts of a TBox of a Telecommunication domain.

In the following both latter levels of knowledge will be required to model our domain:

- $T := \langle \mathcal{T}, \mathcal{A} \rangle$ of the studied domain

wherein $\mathcal{T}$ be the TBox and $\mathcal{A}$ be the ABox.

DLs are a family of logics that were developed for not only modelling complex hierarchical structures and also to provide a specialized reasoning engine to perform inferences on these structures.
Therefore basic inferences on concept descriptions in a TBox (e.g., subsumption checking) and on individuals in an ABox (e.g., instance checking) can be performed.

In the following two sections we focus on TBox standard reasoning inferences such as subsumption, and also on some non standard reasoning. **These inferences will be key in the rest of this dissertation since they will be re-used to perform composition of services.**

$$
\begin{aligned}
&Offer \equiv \forall priceOffer.Price \sqcap \forall interfacedBy.Service \\
&Commercial\_offer \equiv \forall comOffer.Offer \\
&NetworkConnection \equiv \forall netPro.Provider \sqcap \forall netSpeed.Speed \\
&SlowNetworkConnection \equiv NetworkConnection \sqcap \forall netSpeed.Adsl1M \\
&FastNetworkConnection \equiv NetworkConnection \sqcap \forall netSpeed.AdslMax \\
&Speed \equiv \forall\, mBytes.NoNilSpeed \\
&Adsl1M \equiv Speed \sqcap \forall\, mBytes.1M \\
&AdslMax \equiv Speed \sqcap \forall\, mBytes.Max \\
&Max \sqsubseteq 1M \sqsubseteq NoNilSpeed \\
&ZipCode \sqsubset \top, Email \sqsubset \top, Address \sqsubset \top, PhoneNum \sqsubset \top \\
&Invoice \sqsubset \top, DeliveryID \sqsubset \top, Service \sqsubset \top \\
&ZipCode \sqsubseteq \neg Email, Invoice \sqsubseteq \neg Service \\
&IPAddress \equiv Address \sqcap \forall protocol.IP \\
&VoIPId \equiv Address \sqcap \forall network.FTLocal \\
&VideoDecoder \equiv Decoder \sqcap \forall decrypt.Video
\end{aligned}
$$

Figure 1.4: Part of the Terminological Box $\mathcal{T}$ of an $\mathcal{ALE}$ Telecom Ontology T.

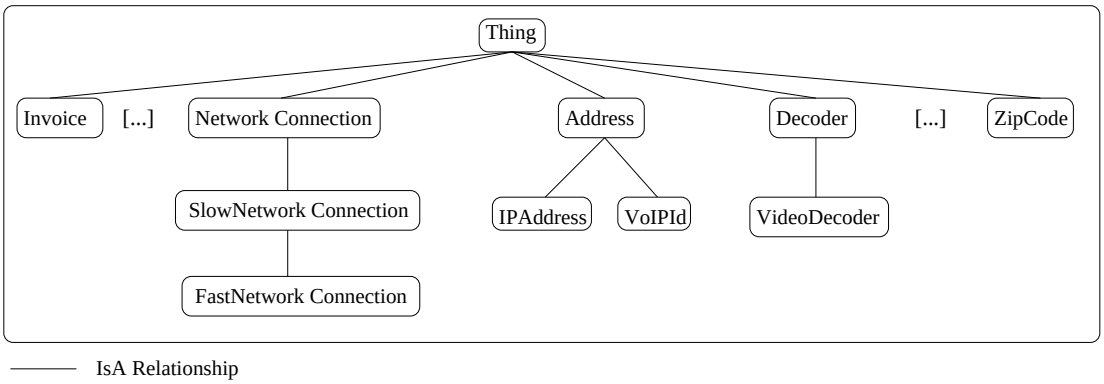


Figure 1.5: A graphical (hierarchy) View of Ontology depicted in Figure 1.4.

1.2.3 Inference in Description Logics

Here we remind standard inferences in DLs, and finally we describe the **notion of difference between descriptions** which is the core operation that we use in our framework.

Standard Inferences

Based on DL semantics and the terminological knowledge $\mathcal{T}$ of a knowledge base, basic DL inferences on $\mathcal{T}$ are as follows: *satisfiability*, *subsumption*, *equivalence* and *disjointness* [58] on

$\mathcal{T}$.

Definition 3. ***(Satisfiability, Subsumption, Equivalence and Disjointness)***

- ***Satisfiability***. *A concept C is satisfiable with respect to $\mathcal{T}$ if there exists an interpretation $\mathcal{I}$ of $\mathcal{T}$ such that $C^{\mathcal{I}}$ is nonempty. In this case we say also that $\mathcal{I}$ is an interpretation of C.*
- ***Subsumption***. *A concept C is subsumed by a concept D with respect to $\mathcal{T}$ iff $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ for every interpretation $\mathcal{I}$ of $\mathcal{T}$. In this case we write $\mathcal{T} \models C \sqsubseteq D$.*
- ***Equivalence***. *Two concepts C and D are equivalent with respect to $\mathcal{T}$ iff $C^{\mathcal{I}} = D^{\mathcal{I}}$ for every interpretation $\mathcal{I}$ of $\mathcal{T}$. In this case we write $\mathcal{T} \models C \equiv D$.*
- ***Disjointness***. *Two concepts C and D are disjoint with respect to $\mathcal{T}$ iff $C^{\mathcal{I}} \cap D^{\mathcal{I}} = \emptyset$ for every interpretation $\mathcal{I}$ of $\mathcal{T}$.*

Such basic inferences are required not only to maintain and to guarantee consisteny of DL knowledge bases but also to classify them. For instance, the TBox classification aims at placing a new concept in the suitable place in a taxonomic hierarchy according to the partial order induced by subsumption relationships among the other defined concepts.

Example 1. ***(Illustration of Standard Inferences)***
With respect to the TBox $\mathcal{T}$ in Figure 1.4, we have the following subsumption relationships:

- $\mathcal{T} \models$ `NetworkConnection` $\sqsupset$ `SlowNetworkConnection`;
- $\mathcal{T} \models$ `NetworkConnection` $\sqsupset$ `FastNetworkConnection`;
- $\mathcal{T} \models$ `Adsl1M` $\sqsubset$ `Speed`;
- $\mathcal{T} \models$ `AdslMax` $\sqsubset$ `Speed`;

Moreover we can infer the following pairs of disjoint concepts:

- $\mathcal{T} \models$ `ZipCode` $\sqcap$ `Email` $\sqsubseteq \bot$;
- $\mathcal{T} \models$ `Invoice` $\sqcap$ `Service` $\sqsubseteq \bot$;

The concept `ZipCode` *is disjoint from concept* `Email` *since* `ZipCode` *is subsumed by the negation of* `Email`.

In litterature of the semantic Web, DLs such as $\mathcal{ALE}$ and $\mathcal{ALN}$ are used to encode domain ontologies since they provide an interesting trade-off between expressivity and complexity. **In the same way we focus on the $\mathcal{ALE}$ and $\mathcal{ALN}$ DLs to encode our ontologies.**

On contrary, the semantics of unrestricted RDF-S and OWL-Full is non-traditional and the reasoners built for OWL Full fragment tend to be sound but incomplete. Indeed there is no straight-forward way to extend the existing reasoners to support the full expressivity of OWL-Full.

Although the previous standard inferences help structuring a knowledge base e.g., by automatically building a (consistent) concept hierarchy, there are still some limitations. Indeed the standard reasoning are, for example, not appropriate when it comes to (automatically) generating new concept descriptions (e.g., least common subsumer [47], most specific concept [18], concept difference [35, 198], concept abduction [55]) from given ones. Therefore, besides the standard inferences, additional, so called non-standard inferences, are required [102].

A Non Standard Inference in Description Logics: The Difference Operation

In this section, we focus on one category of non standard inferences i.e., the difference operation between two concept descriptions. More specifically we investigate on the definitions of semantic difference [198], syntactic difference [35] and concept abduction [48, 55, 49] in DLs.

The semantic difference, first introduced by [198], enables us to remove from a given description all the information contained in another description. The difference between two concept descriptions C and D with $C \sqsubseteq D$ is given by the following definition.

Definition 4. ***(Semantic Difference)***
Let C, D be two concept descriptions with $C \sqsubseteq D$. The semantic difference $C - D$ of C and D is defined by:

$$C - D := \max_{\sqsupseteq}\{B | B \sqcap D \equiv C\} \tag{1.1}$$

Roughly speaking, the latter difference is defined as being a description containing all information which is a part of the description C but not a part of the description D. This definition of difference operation requires that the second operand subsumes the first one. However, in case the operands C and D are incomparable with respect to the subsumption relation (i.e., C is not subsumed by D), then the difference $C - D$ can be given by constructing the least common subsumer of C and D, that is, $C - D := C - lcs(C, D)$. It is worth noting that, in some description logics, the set $C - D$ may contain descriptions which are not semantically equivalent.

Example 2. ***(Redundancies in Semantic Difference [198])***
Consider the $\mathcal{ALN}$ descriptions $C \doteq (\forall R.P_1) \sqcap (\forall R.\neg P_1)$ and $D \doteq (\forall R.P_2) \sqcap (\forall R.(\leq 4S))$. The set $C - D$ includes the non equivalent descriptions $(\forall R.\neg P_2)$ and $(\forall R.(\geq 5S))$.

Even if [198] provides sufficient conditions (i.e., structural subsumption relation) to characterize the uniqueness of difference, some TBoxes cannot be considered. [36, 35, 102] proposed a refinement of the previous definition 4 by taking the syntactic minimum (w.r.t a subdescription ordering $\preceq_d$ [19, 36, 35, 102]) instead of a semantic maximum.

In this way [35] used the subdescription ordering $\preceq_d$ to deal with syntactical redundancies and then find a compact representation of the difference of two concepts in $\mathcal{ALC}$ or $\mathcal{ALE}$. Moreover they defined the difference between two (in)comparable concept descriptions C and D.

Definition 5. ***(Syntactic Difference [35])***
Let C, D be two concept descriptions. The syntactic difference $C \backslash D$ of C and D is defined as the syntactic minimum with respect to the subdescription ordering $\preceq_d$.

$$C \backslash D := \min_{\preceq_d}\{B | B \sqcap D \equiv C \sqcap D\} \tag{1.2}$$

Intuitively, the idea is to remove all subdescriptions from C which are either redundant in C or already present in D. It should be noted that in case of $C \sqsubseteq D$, and thus, $C \sqcap D \equiv C$, the only difference to [198]'s difference operator is that the minimum with respect to $\preceq_d$ is used instead of the maximum with respect to $\sqsubseteq$.

Finally, it should be noted that a priori the syntactic difference between C and D is also not uniquely determined. By abuse of language and notation, we will still refer to the difference $C \backslash D$.

Example 3. ***(Syntactic Difference Illustration)***
Let $SlowNetworkConnection$ and $NetworkConnection$ be two $\mathcal{ALE}$ descriptions illustrated in Figure 1.4. According to Definition 5, the description missing in $NetworkConnection$ to be $SlowNetwork$-Connection is referred by

- $SlowNetworkConnection \backslash NetworkConnection := \forall netSpeed.Adsl1M.$

Even if Definition 4 captures the real semantic difference between two concept descriptions, Definition 5 has two main advantages. Firstly it does not contain redundancies in its result and secondly it is more readable by a human user.

Considering an $\mathcal{ALN}$ DL, Concept Abduction [48, 55] is also able to compute a concept expression B representing what is underspecified in D in order to completely satisfy C taking into account the information modelled in a TBox $\mathcal{T}$.

Definition 6. ***(Concept Abduction Problem)***
Let $\mathcal{L}$ be a DL, C, D be two concepts in $\mathcal{L}$, $\mathcal{T}$ be a set of axioms in $\mathcal{L}$ and $\mathcal{A}$ be a set of assertions. A Concept Abduction Problem (CAP), denoted as $\langle \mathcal{L}, D, C, \mathcal{T} \rangle$ consists in finding a concept $B \in \mathcal{L}$ such that $\langle \mathcal{T}, \mathcal{A} \rangle \models D \sqcap B \sqsubseteq C$.

Concept Abduction may appear similar to Concept Difference [198], yet it is not so. Indeed performing a difference operation requires a subsumption relation between descriptions to be matched since Concept Difference 4 requires the extra strict condition $C \sqsubseteq D$. This strict condition may make Concept Difference 4 hard to use in a matchmaking process (e.g., Intersection match level), where descriptions overlap is usually a sufficient condition to start the process. However Concept Difference 4 and 5 are more accurate than Concept Abduction since 4 and 5 perform an equivalence between two concept descriptions ($\langle \mathcal{T}, \mathcal{A} \rangle \models B \sqcap D \equiv C$ or $B \sqcap D \equiv C \sqcap D$) whereas the Concept Abduction computes a subsumption of concept descriptions ($\langle \mathcal{T}, \mathcal{A} \rangle \models B \sqcap D \sqsubseteq C$).

In the following we suggest to use definition 5 since

i) its result will be reused by human user in our semantic web service composition framework,

ii) its result is more accurate than Concept Abduction,

iii) its result does not contain redundancies,

iv) it is defined for incomparable descriptions without major change of the definition,

v) it is also defined for $\mathcal{ALE}$ descriptions logics.

1.2.4 Synthesis

The semantic Web has the potential to provide the web services infrastructure with the semantic information that it needs. It could provide formal languages and ontologies to reason about service descriptions, message content, business rules and relations between these ontologies. In this way, the semantic Web, its underlying formal logic and web services are synergistic: the semantic web transforms the Web into a repository of computer readable data, while web services provide the tools for the automatic use of that data.

1.3 Semantic Web Service

Semantic web services result from the integration of semantic metadata, ontologies, formal tools and the web services infrastructure.

A semantic web service is described as a web service whose internal and external description is in a language that has well-defined semantics [197]. The result of using the semantic web is an

unambiguous description of the interface of the Web service which is machine understandable and provides the basis for a seamless interoperation among different services. Specifically, semantic web services rely on the semantic web to describe:

- *rich functionality* of *functional level based Web services* such as:
  - their functional operations and parameters;
  - the content of the messages that they exchange.
- *behaviour* of *process level based Web services* such as:
  - the order of the messages exchanged;
  - the state transitions that result from such exchanges.

Figure 1.6 illustrates these two levels of semantics in an extended Web service specification stack.

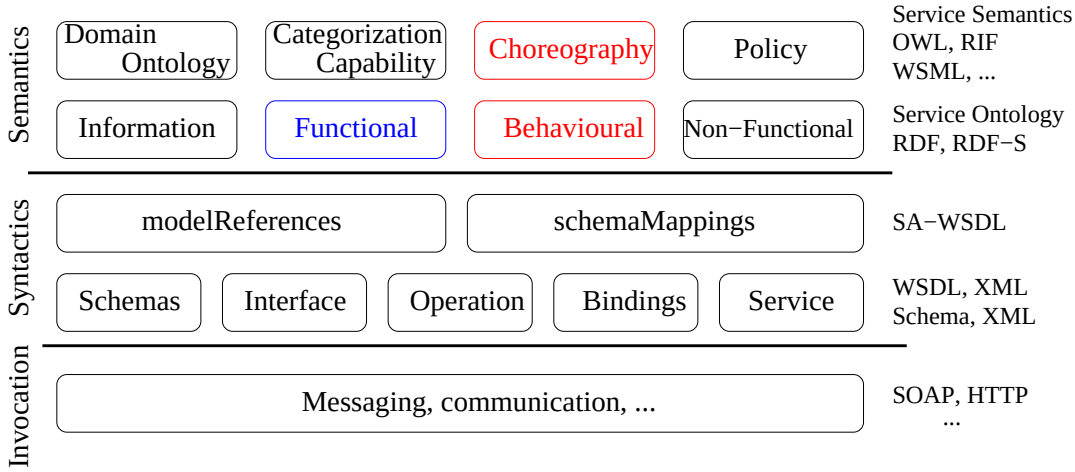


Figure 1.6: Process and Functional Level Description of Semantic Web Services in an Extended Service Specification Stack.

Therefore, description of semantic Web services is unambiguously computer interpretable, and facilitates maximal automation and dynamism in web service discovery [24], selection [208], composition [175], choreography [66], orchestration [32], invocation, monitoring, management, recovery and compensation.

In parallel, the use of the semantic web to describe web services has wide ranging consequences. It allows the description of additional properties of web services, such as the quality of service and security constraints in a coherent and uniform way that is universally understood. Furthermore, and most importantly, the description of the states produced by the execution of the web service is the basis for the description of its capabilities as a transformation from its input parameters and preconditions, to its output parameters and effects.

In the rest of this section, we first focus on semantic Web service at functional level. Then we present the standard language proposals used to describe semantic web services.

1.3.1 Semantic Web Service Description at Functional Level

At functional level, semantic Web services are described in the same way as Web services i.e., *service name*, *operations names*, *message names* (also known as input and output messages/parameters), *interface name*. The major change and benefit is about their description:

> *In the semantic Web, input and output parameters of Web services are described by means of DL descriptions in a domain ontology.*

The latter parameters (i.e., inputs and outputs) characterize respectively knowledge preconditions and knowledge effects of executing web services[5]. In contrary effects characterize physical effects of executing services whereas preconditions characterize the conditions under which a particular service may be executed.

Input and Output Parameters of Web Services

In the functional level of semantic Web service description, all input and output parameters of semantic Web services refer to concepts in the TBox $\mathcal{T}$ of a domain ontology T. In other words syntactic Web services have been enhanced with semantic annotation of their functional parameters.

Example 4. ***(Some Web Services at Functional Level)***
In the following example, we consider six Web services[6] defined at functional level i.e., input and output parameters are annotated by means of concepts in the TBox $\mathcal{T}$ (1.4) of a domain ontology T.

- *`AdslEligibility-`, `AdslEligibility` and `AdslEligibility+`, that, starting from a `PhoneNum`, a `ZipCode` and an `Email` address, returns respectively the `SlowNetworkConnection`, `NetworkConnection` and `FastNetworkConnection` of the desired zone;*
- *`VoiceOverIP`, that, starting from a `PhoneNum` and a `SlowNetworkConnection`, returns the `VoIPId` of the ADSL line a Telecom operator needs to install the line;*
- *`TvOverIP`, that, starting from a `PhoneNum` and a `FastNetworkConnection`, returns a serial number of a `VideoDecoder` required to access video over IP;*
- *a `LiveBox` service returns the `Invoice` of the commercial offer the user requested, depending on a `PhoneNum`, `IPAddress` and serial number of a `Decoder`.*

Such services require and provide some information. On the one hand Web services require some instances (defined in $\mathcal{A}$) of their input parameters (defined in $\mathcal{T}$) to be executed. On the other hand Web services return some instances (defined in $\mathcal{A}$) of each output parameters of the latter services.

Example 5. ***(A Semantic Web Service at Design and Run Time)***
Suppose `AdslEligibility` be one of the Web services illustrated in the previous example. Such a Web service (Figure 1.7(a)) is semantically enhanced by means of semantic annotation of its functional parameters:

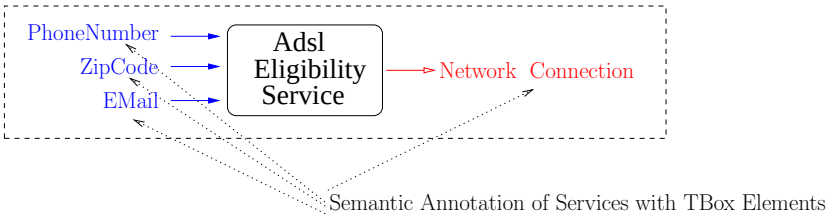
1. *input parameters i.e., `Phone Number`, `ZipCode`, `EMail`;*

[5] In the rest of the Ph.D report, we assume without loss of generality that each Web service (at functional level) have one operation. However we keep in mind that Web services may have more than one operation in general.

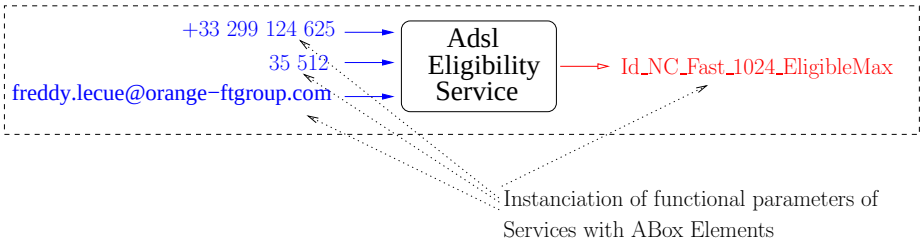
[6] The six Web services described in this Chapter represent a sample of the set of 35 Web services defined in the Telecommunication scenario in Chapter 6.

2. *output parameters i.e.,* `Network Connection`,

the whole with respect to the domain ontology T and TBox $\mathcal{T}$ (Figure 1.4). Once its execution is performed we obtain an instance of `Network Connection` *which is in the ABox i.e.,* `Id_NC_Fast_1024_EligibleMax` *(Figure 1.7(b)). To this end the three input parameters of the service have been first instantiated.*



(a) Semantic Description of a Web Service.



(b) Instanciation of a Web Service at Run Time.

Figure 1.7: Illustration of a Semantic Web Services.

Roughly speaking, given a set of instances in an ABox, semantic Web services are able to generate some instances of (potentially new) concepts, depending on their semantic definitions.

Preconditions and Effects of Web Services

In addition to input and output parameters, preconditions and effects can be required by some Web services. The latter parameters are modelled as conditions on input and output parameters.

Example 6. ***(Preconditions of Semantic Web Services)***
Suppose Example 4. The `AdslEligibility-`, `AdslEligibility` *and* `AdslEligibility+` *services require*

- `PhoneNum` *ph to be a French phone number i.e.,* $FrenchPhone(ph)$;
- `ZipCode` *zc to be a French zip code i.e.,* $FrenchZip(zc)$;
- `Email` *address to be a valid address i.e.,* $validMail(e)$.

before execution. Same category of preconditions can be added to preconditions of Web services presented in Example 4.

Example 7. ***(Effects of Semantic Web Services)***
Suppose Example 4. The `AdslEligibility` *service returns a* `NetworkConnection` *nc of the desired zone, which is valid after execution i.e., validNC(nc). Same category of effects can be added to preconditions of Web services presented in Example 4.*

Such parameters respectively further restricts the domain on input and output parameter of Web services. Therefore we limit the set of instances in an ABox that can meet the constraints of Web services parameters.

Information-Providing and World-Altering Services

In this Ph.D report we will focus on two classes of Web services i.e., **information-providing services** and **world-altering services**.

Definition 7. (Information-Providing Service)
An information-providing service (aka sensing or information-gathering action in AI planning) is described by means of only input and output *parameters.*

Definition 8. (World-Altering Service)
A world-altering service requires preconditions and provides side effects.

Roughly speaking world-altering services are services that change the state of the world (e.g., any buying service) and information-providing services are services that provide some actions of sensing to gather relevant information (e.g., a weather forecast service).

1.3.2 Some Standard Proposals

The field of semantic web service, which got under way around 2001 [200, 140], includes substantial bodies of work, such as the efforts around:

- an *ontology for Web services*:
  - *OWL for Services* i.e., OWL-S (formerly DAML-S) [12];
  - the *Semantic Web Services Ontology* i.e., SWSO [179];
  - the *Web Services Modeling Ontology* i.e., WSMO [60];
- an *extension of a Web service standard*:
  - the *Semantic Annotation for WSDL* i.e., SA-WSDL (formerly WSDL-S) [192, 59].

These proposals of semantic web service standard attempt to close the gap between the semantic web and the web services infrastructure.

This section focuses its attention on the previous efforts and their relation. However other approaches with more limited focus can be found, for instance, among the submissions to *Semantic Web Service* Challenge [7]. In the rest of this section, we *briefly* describe and give some relevant references on the most complete framework i.e., OWL-S. Then SWSO and WSMO, are described and compared with OWL-S. Finally SA-WSDL[8] is presented. Table 1.2 summarizes the *functional*, *process* and *invocation* levels used to describe Web service in the latter standard proposals.

[7] http://sws-challenge.org/

[8] **SA-WSDL and WSMO are the languages we used to describe semantic web services** involved in industrial application such as in France Telecom R&D. However it is straightforward to interchange the semantic web service language we use without loss of generality

	Ontology for Web Services						Web Service Standard Extension	
	OWL-S		WSMO		SWSO		SA-WSDL	
Functional	Service Profile	IOPEs Service Category Non Functional Properties	Service Capability	IOPEs Non Functional Properties	Service IOPEs	IOPE Message Type Channel (Com.)	Internal, External Semantic Annotation	IO (model reference, Schema Mapping)
Process	Service Model, Process ***	Atomic Proc. Simple Proc. Composite Proc.	Service Choreography and Orchestration **	state := Ontology Guarded Transition (If_Then_Else rules)	Process Model *****	Atomic Process Composite Process	✗	
Inv.	Service Grounding	Binding to WSDL, SOAP	Service Grounding	Binding to WSDL, SOAP	Service Grounding	Binding to WSDL, SOAP BPEL	Binding to WSDL, SOAP	
Execution Env.	✗		WSMX		✗		METEOR-S	

Table 1.2: Fucntional, Process, Invocation Levels and Execution Environment in OWL-S, WSMO, SWSO, SA-WSDL. Legend: ✗ = not supported, * = level of expressivity.,Inv. = Invocation, Env = Environment.

OWL-S

The first major ontology for describing web services is OWL-S[9] [12]. As ontology, OWL-S is based on OWL to define the concept of web service within the semantic web. In addition it provides a language to describe semantics of actual web services that can be discovered and then invoked using standards such as WSDL and SOAP. OWL-S uses the semantic annotations and ontologies of the semantic web to relate the description of a Web service, with descriptions of its domain of operation.

OWL-S requires that web services be represented by a specification of their *capabilities*, a representation of their *interaction protocol* and a specification of *how to compile the actual messages to exchange*.

To this end, an OWL-S web service is defined as a OWL class with three properties which relate the Web service to the i) *service profile*, ii) the *service model process* and iii) the *service grounding*.

The *service profile* provides a representation of the capabilities of the web service in terms of the input, preconditions and output, effects transformation that it produces and of a set of non-functional parameters that specify availability, quality and other properties of the service.

The *service model process* provides a detailed view of process of the web service from which the requester can derive the interaction protocol with the provider.

Finally *the grounding* maps the process model into a WSDL specification of how to interact with the web service. OWL-S reliance on WSDL provides the bridge between the semantic web and the web services infrastructure.

[9] http://www.w3.org/Submission/OWL-S/

SWSO

The *Semantic Web Services* Framework (SWSF)[10] [179], initiated by the *Semantic Web Services* Initiative, builds loosely on OWL-S to provide a more comprehensive framework, in the sense of defining a larger set of concepts. SWSF specifies a Web-oriented language, SWSL, with a logic programming layer and also a first-order logic layer. Since SWSO is completely axiomatized in first-order logic, it avoids the need to use hybrid (DL-based and FOL-based) reasoning as is the case with OWL-S.

In such a framework, Web services are specified by means of three classes i.e., the *service IOPEs*, the *process model* and the *service grounding*.

It uses SWSL to define an ontology of service concepts (SWSO), and takes advantage of SWSL's greater expressiveness (relative to OWL) to more completely axiomatize the concepts. In the same way as the *service profile* of OWL-S, the *service IOPEs* defines IOPEs as concepts of an ontology. However this level is more expressive since message type exchanged by Web service can be given.

The *process model* od SWSO is built on a mature pre-existing ontology of process modelling concepts, the Process Specification Language (PSL) [42]. This specification is more espressive than the *service model process* of OWL-S. For instance *OccActivity* that satisfies the occurrence constraints on subactivities, *TriggeredActivity* as activity which occurs whenever a state condition is satisfied, can be both expressed in SWSO.

The essence of the *service grounding* approach of SWSO is to establish a mapping between selected message-oriented constructs in SWSO and their counterparts in WSDL service descriptions. An extension to BPEL4WS[11] [10] is also possible.

WSMO

The Web Services Modeling Ontology (WSMO)[12] [60] shares with OWL-S and SWSF the vision that ontologies are essential to support, for instance, automatic discovery, inter-operation, composition of *Web Services*. Therefore, as OWL-S and SWSO, WSMO is an ontology for describing various aspects related to semantic web services. Moreover, in the same way as SWSF, the WSMO effort defines an expressive Web-oriented language, WSML [107], which provides a uniform syntax for sub-dialects ranging from description logic to first-order logic.

Like OWL-S and WSMO, Web services are specified by means of three classes i.e., the *service capability*, the *process choreography, orchestration* and the *service grounding*.

WSMO declares the *service capability* i.e., inputs, outputs, preconditions, and results (although using some-what different terminology) associated with services

Unlike OWL-S and SWSO, WSMO does not provide a notation for building up composite processes in terms of control flow and data flow. Instead, it focuses on specification of *internal and external choreography and orchestration* using an approach based on abstract state machines (with guarded transitions).

The *service grounding* of WSMO is defined in the same way as the one of OWL-S. This task is achieved by mediator, which is a key concept in WSMO. In WSMO's approach, mediators perform tasks such as translation between ontologies, or between the messages that one web

[10] http://www.w3.org/Submission/SWSF-SWSO/

[11] BPEL4WS provides a language for the formal specification of business processes and business interaction protocols. By doing so, it extends the Web Services interaction model and enables it to support business transactions.

[12] http://www.w3.org/Submission/WSMO-primer/

service produces and those that another web service expects. WSMO includes a taxonomy of possible mediators that helps to classify the different tasks that mediators are supposed to solve. The definition of mediators in WSMO calls attention to some important translation tasks associated with web services. Not surprisingly, these same translation tasks are needed in support of interactions with OWL-S described *Web Services*. Some OWL-S based systems [154, 153, 99] also make use of mediator components. However, rather than requiring the existence of a distinguished type of entity in the *Web Services* infrastructure, OWL-S takes the view that mediators are services and as such these mediation services can use the mechanisms provided by OWL-S for discovery, invocation and composition.

Other distinguishing characteristics include WSMO's emphasis on the production of a reference implementation of an execution environment, WSMX, and on the specification of mediators (i.e., mapping programs that solve the interoperation problems between Web Services).

SA-WSDL (formerly WSDL-S)

Recently the W3C produced a standard set of "Semantic Annotations for WSDL and XML Schema" (SA-WSDL). Contrary to OWL-S, SWSO and, WSMO, SA-WSDL is not defined as an ontology for Web services. SA-WSDL[13] [192, 59, 101] is a set of proposed extensions for WSDL, by which semantic annotations may be associated with WSDL elements such as operations, input and output type schemas, and interfaces.

> *SA-WSDL aims to support the use of semantic concepts analogous to those in OWL-S while being agnostic to the semantic representation language [59].*

The way in which SA-WSDL allows one to specify a correspondence between WSDL elements and semantic concepts is very similar to how the OWL-S grounding works. Indeed, something very much like OWL-S declarations could be used as the referents of the SA-WSDL attributes.

In the same way as WSMO, the former version of SA-WSDL i.e., WSDL-S [3] can be associated with an execution environment and tools known as METEOR-S[14] [163], but comparison with tools and environments is beyond the scope of this Ph.D thesis.

As suggested by [131], using SA-WSDL with OWL-S and other previous semantic web services frameworks as the source of annotation referents is a direction to couple strength of OWL-S and weakness of SA-WSDL. The most notable difference between SA-WSDL and the OWL-S grounding is that with SA-WSDL the correspondence must be given in the WSDL document, whereas with OWL-S it is given in a separate OWL document. Thus, the aims of SA-WSDL are compatible with those of OWL-S, but SA-WSDL focuses on the practical advantages of a more lightweight, incremental approach.

Some Alternative Proposals

Alternative languages for semantic web services such as BPEL4SWS [65] (i.e., a semantic adaptation for BPEL), SPATEL [86] (i.e., an naive alternative to OWL-S), USDI [21] are still under development.

[13] A W3C recommendation since August 2007, the 28th. http://www.w3.org/TR/2007/REC-sawsdl-20070828/
[14] http://lsdis.cs.uga.edu/projects/meteor-s/

Discussion

SA-WSDL is the first step toward standardizing semantic web service. It forms the basis for interoperation between the various semantic web service efforts that previously could not seem to find any common ground. As we have shown, SA-WSDL itself is not a complete technology for allowing automation. Indeed we have to provide a service ontology and the appropriate domain ontologies to describe web services. The major semantic web service frameworks (OWL-S, SWSO and WSMO) have already started to embrace SA-WSDL for grounding (connecting the semantic framework to the WSDL descriptions of web services). The next step is to rework these frameworks with SA-WSDL in mind, refactoring them into parts that can be attached using SA-WSDL annotations. Such a refactored framework based on RDF and OWL would likely be the basis for further standardization in the W3C.

1.4 Discussion

Services technologies are being shaped by, and increasingly will help shape, modern society as a whole, especially in vital areas such as dynamic business, health, education and government services. Applying services technologies leads to reduced complexity and costs, exposing and reusing core business functionality, increased flexibility, resilience to technology shifts and improving operational efficiency.

For all these reasons, it is expected that the *Service Oriented Computing* paradigm will exhibit a steeper adoption curve, as it solves expensive and intractable business and technology problems, and will infiltrate more of the applications portfolio, than previous application technologies.

As previously said, key to developing service-based applications is the SOA and more specially web services. Currently, the SOA provides the basic operations necessary to describe, publish, find and invoke services. However, those basic operations while they help services to be ubiquitous and universal are not a complete solution.

The AI and semantic Web community has concentrated their efforts in giving richer semantic descriptions of web services that describe the properties and capabilities of web services in an computer-interpretable form.

Indeed, for services to be used widely, these additional functionalities have to be considered for achieving (in parts) dynamic and automated service discovery, composition, and invocation of services over the Web. Towards these issues, semantics seems an interesting investigation field to overcome the latter issues. Such concerns are addressed by the semantic web service initiative, which targets the use of formal languages for semantically describing all relevant aspects of Web services.

However, it is stems that actual work on semantic web services does not yet overcome the latter challenges. On the one hand, there are quite a few challenging problems for the near future as regards the service specification. On the other hand, even if standards for semantic web services are considered there are still notable challenges including, for instance, dynamic service discovery, composition, execution, and management.

Chapter 2

Web Service Composition

The full potential of web services as a means of developing dynamic e-Business solutions will only be realised when applications and business processes will be able to integrate their complex interactions into composite added value services. Web services technologies offer a viable solution to this problem since they support coordination and offer an asynchronous and message oriented way to communicate and interact with application logic. Therefore, automated Web service composition has attracted great interest in the research community.

In our work we will focus on goal-oriented Web service composition. The latter goals is defined according to Definition 9.

Definition 9. ***(Goal of a Composition)*** *Given a TBox $\mathcal{T}$ of a domain ontology T and its ABox $\mathcal{A}$, a goal of a Web service composition is defined by a set of n concept descriptions $C_{i,1\leq i\leq n} \in \mathcal{T}$. Such a goal consists in discovering by composition of Web services new ABox elements which are instances of each C_i.*

According to Section 1.1.3, Web service can be described at two different levels i.e., the *functional* and *process* level. Therefore these latter levels provide different functionalities to support efficiently their level of composition i.e.,

- *functional level composition*;
- *process level composition.*

Indeed the process related to their composition varies from one level to another one since different description requirements, results and issues are involved.

In the same way as both levels of Web service description, both previous levels of composition are complementary, and then can be combined. For instance Web service composition can be achieved by first a level of functional composition, and then a level of process level composition. That is why a description and main references on *process level of composition* models are briefly reviewed in this Chapter. These models would be selected, depending on the user requirements, to couple and combine any one of them with our *functional level based composition* (see Chapters 3, 4, and 5).

This chapter surveys a representative set of existing literature that is related to the work presented in this thesis. The chapter is divided into five main sections. The first section defines

web service composition at *functional level* and provides a literature review of the state-of-the-art i.e., the most closely related work to our Ph.D studies. Moreover this section identifies the main challenges towards automation of Web service composition. The second section defines web service composition at *process level* and briefly focuses on related work. In section 2.3 we focus on the complementarity feature of the two different approaches. Section 2.4 briefly overviews two methods to model web service composition i.e., Orchestration and Choreography. Finally section 2.5 draws some conclusions.

2.1 Functional Level Composition

In this section, we first describe *functional level composition* (henceforth FLC) in details and focus on two key concepts to achieve such a level of composition i.e., *semantic dependences* and *causality relationships* between services. Then we will review related works of two main categories of compositions, which are roots of FLC:

- *Matchmaking and semantic dependences based composition*;
- *AI planning and causality relationships based composition.*

Each model together with their advantages and limitations will be studied in a detailed and critical analysis.

2.1.1 Description

Functional Level Compositions, also known as data flow driven compositions [104, 197, 154, 52, 126] consider composition of web services which are described on a functional view.

As highlight in Section 1.1.3 *functional level* based Web services are restricted to atomic interactions i.e., black boxes which are defined only in terms of their input and output parameters as well as of their preconditions and effects. In this direction FLC addresses the following issue:

> *FLC addresses the problem of selecting a set of web services that, combined in a suitable way, are able to perform a composition goal.*

In such a context the composition goal is defined as the overall functionality that the composed service should implement, again in terms of its inputs, outputs, preconditions, and effects (i.e., functional level).

By considering such a level of description, FLC is very much related to traditional AI planning [187, 52]. The available web services correspond to planning operators that require certain input parameters and preconditions and provide some output parameters and effects. The planning results is a partial ordering of web services arranged in a simpler version of a workflow in order to fulfil the composition goal. In such approaches the resulting partial ordering of services consists of:

- *causal laws* between some of its services.

The causal laws are not only i) *causality relationships* between effects and preconditions of services (also known as *causality relationships* between services) but also ii) complex relationships between services (e.g., relationships that link an input parameter of a service to an output parameter of another service under some conditions). Therefore they first ensure that preconditions

of services are satisfied before any service execution. Moreover they add an explicit and complex relationship between parameters of different services.

Unlike approaches which have roots in AI planning, other approaches re-use semantic description of web service to perform automated composition. In such approaches the resulting composition consists of

- *semantic dependences* between some of its services. These dependences can be inferred through *semantic matchmaking*.

The semantic dependences link functional parameters of services through their output and input parameters. In other words these dependences are required to semantically link output to input parameters of Web services. The semantic dependences between web services are considered as an important issue to form new value-added services at functional level. In this spirit, [52] combine Web service discovery and FLC together, as complementary processes, to find a chain of suitable web services.

The Figure 2.1 illustrates an *ideal functional composition* of Web services that assumes semantic descriptions on input, output parameters together with preconditions and effects on Web services' parameters.

The functional description of web services and composition goals can be provided, for instance, by means of some standard proposals such as the OWL-S *Service Profile* [12], WSMO *Service Capability* [60], or SA-WSDL specification.

2.1.2 Matchmaking and Semantic Dependences based Composition

In this section we first present some backgrounds on matchmaking of semantic web services i.e., a key concept to compute compositions of services wherein *semantic dependences* are involved. From this definition, we review main related work in *matchmaking and semantic dependences based Composition* and closest to our approach i.e., [174], [188], [105] and [15].

These approaches adapt semantic matchmaking for retrieving composition by means of *semantic dependences* between output and input parameters of web services.

Concepts and Definitions

Here, we will describe how the existing work on DL reasoning and semantic matchmaking have been used to facilitate semantic web service matchmaking i.e., a key issue in FLC. From this, we will discuss how the matchmaking task is related to FLC and then explore these functional parameters matchmaking based-FLC in details.

a) Matchmaking and Semantic Web Service Discovery:

Matchmaking on semantic web services has been first primarily studied in the field of service discovery. Given a set of service advertisements S_a and a service request S_r, a discovery process aims at computing subsumption relation between the previous services S_a and S_r in order to retrieve the most *close* service of S_r (Figure 2.2(a)). The subsumption relation is valued between functional parameters of Web services, especially their input and output parameters, which are defined as concept (using DL descriptions) of a domain ontology $\mathcal{T}$.

The matchmaker, introduced by [152], is the first system that implemented this idea of semantic based Web service discovery.

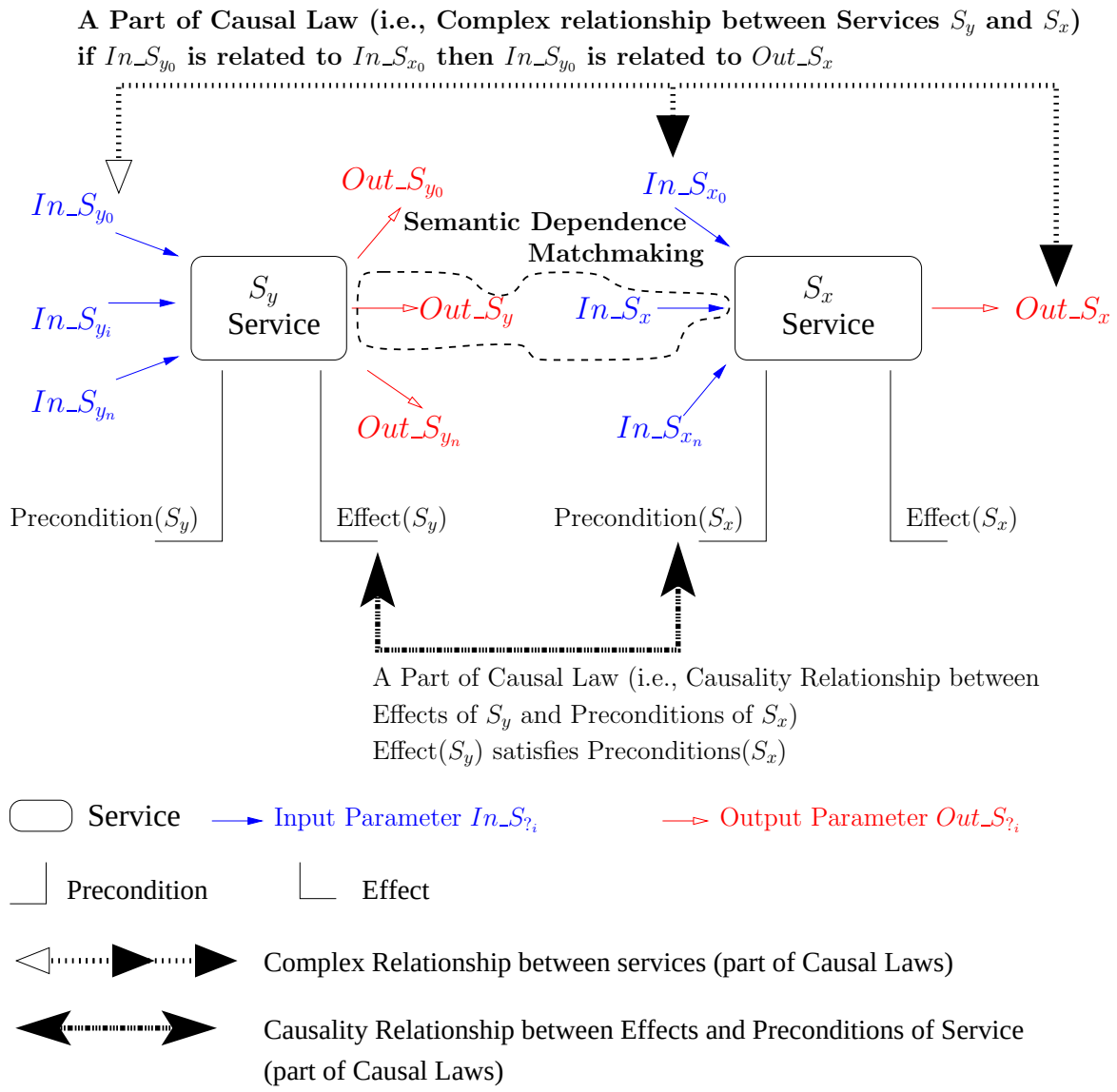


Figure 2.1: An Ideal Functional Level Composition of two Web Services with Semantic Dependences and Causal Laws.

This matchmaker system uses OWL-S profiles to describe service requests as well as the services advertised. From this, a service provider publishes an OWL-S description to a common service repository and a service requester specifies a service profile, still in OWL-S, for the desired service she wants to dynamically retrieved.

The request profiles S_r are then matched by the service registry to advertised profiles S_a using DL reasoning (standard DL reasoning such as subsumption in the first version of the matchmaker) as the core inference service. In particular, the matchmaker computes subsumption relations

between each individual input, output parameter of the request S_r and the advertisement service profile S_a.

If the DL concepts of the corresponding parameters are equivalent, there is an **Exact** and thus best match. If there is a subsumption relation between S_a and S_r, there is i) a **PlugIn** in case S_r is more specific than S_a, and a ii) **Subsume** in case S_a is more specific than S_a. Otherwise there is no subsumption relation, then there is no match i.e., **Disjoint**.

More formally, the basic relations used in the matchmaker of [152] are as follows:

- **Exact**. If advertisement S_a and request S_r are equivalent concepts, it is called an Exact match; formally, $\mathcal{T} \models S_r \equiv S_a$.
- **PlugIn**. If request S_r is sub-concept of advertisement S_a, it is called a PlugIn match; formally, $\mathcal{T} \models S_r \sqsubseteq S_a$.
- **Subsume**. If request S_r is super-concept of advertisement S_a, it is called a Subsume match; formally, $\mathcal{T} \models S_a \sqsubseteq S_r$.
- **Disjoint**. Otherwise, S_a and S_r are incompatible; formally $\mathcal{T} \models S_r \sqcap S_a \sqsubseteq \bot$.

Another efficient and less basic approach consists in valuing the **Intersection** matchmaking [124] between two service profiles. For example, in case the subsumption relation between the advertisement and request profiles is not satisfiable, the **Intersection** matchmaking may be assigned when their DL intersection is not empty, In other words advertisement and request descriptions are not disjoint and relaxing some of the constraints on the request may provide better results. Therefore we may add:

- **Intersection**. If the request S_r has common descriptions advertisement S_a, it is called a Intersection match; formally, $\mathcal{T} \not\models S_r \sqcap S_a \sqsubseteq \bot$.

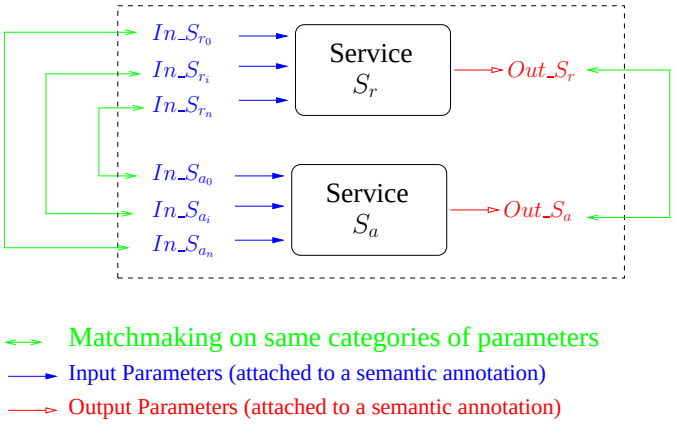
Even if the latter matchmaker is used for discovery purposes by trying to find a semantic relation between services (i.e., semantically close services), it can also be adapted for achieving *semantic web service composition at functional level*.

b) Matchmaking and Functional Level Composition

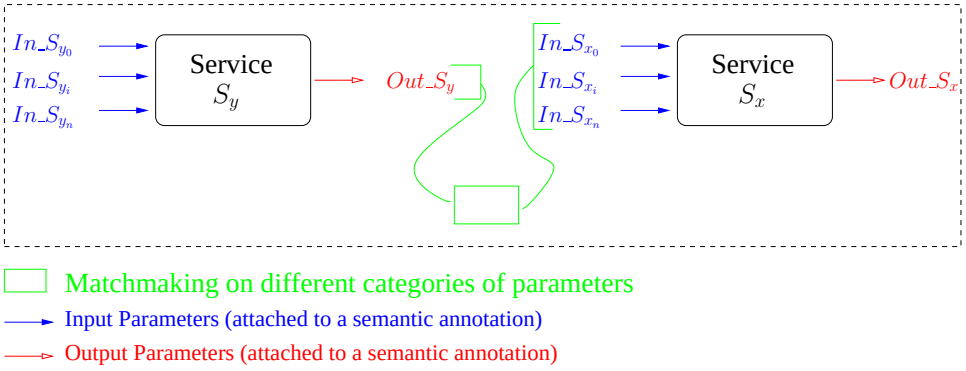
In Web service discovery, matchmaking is performed on same categories of parameters i.e., pair of inputs, or pair of outputs. On contrary functional level composition of Web services can be addressed, by studying matchmaking on pairs of distinct output and input parameters of different Web services S_y and S_x (Figure 2.2(b)). More specifically, the approach consists in computing a set of pairs as matchmaking tasks to retrieve semantic similarities and dependences between such parameters of Web services.

From these dependences and given a set of relevant Web services, a process is required to compose them in order to achieve a given goal e.g., an abstract service defined in terms of input and output parameters.

In the next section we will focus on related work of web service composition that performs FLC, mainly, by means of *semantic dependences* and *matchmaking* between their functional output and input parameters. The following four works have been studied in details for this main reason.



(a) Matchmaking for Discovery.



(b) Matchmaking for FLC.

Figure 2.2: Matchmaking for Discovery and FLC.

A Logic based Approach

Towards the issue of FLC, [174] (also discussed in [175]) introduces a method for automatic composition of semantic web services using **Linear Logic theorem proving** as a **composition mechanism**.

On the one hand, Linear Logic is used to formally define **static definition of web services** i.e., functional and non functional parameters (e.g., qualitative and quantitative values) of web services. *They restrict the expressivity of functional parameters of services to be inputs and outputs, but not preconditions or effects.* On the other hand, Linear Logic, as a logic for specifying concurrent programming, has close relationship [1, 22] with π-calculus (i.e., the formal foundation of many Web service composition languages). Such a logic provides higher **expressive powers in the modelling of dynamic behaviour** (mainly **sequence** and non **concurency**) of composite web services than classical logic. *However conditional compositions of Web services are not supported in Linear Logic.* Indeed their services consisted only of determinist output parameters.

From the service description they elaborate Linear Logic extra logical axioms for each service. These axioms describe the functional description of services in terms of their input and output parameters.

Besides the latter axioms, [174] extends their initial Linear Logic theory by introducing additional axioms related to **semantic similarity between output and input parameters** of services. *In their approach the matchmaking functions are restricted to the subsumption relation i.e., Exact, PlugIn, Disjoint to value this similarity.* Such similarities define their **composability criteria** (relations between Web services in a composition) in their model. To this end they exploit a **semantic reasoner** to detect the subsumption relationships between functional parameters at **design time** i.e., DAML concepts in their approach.

Such relationships are defined as axioms of their model, and can be represented by Linear Logic implication, thus they can be asserted into their Linear Logic theorem prover as axioms. This ensures the **flexibility** of the model.

By considering the two latter axioms their approach is twofold: i) first the process model of a composite service can be generated directly from the complete proof, and ii) in case all subsumption relations have been detected by the semantic reasoner, the web service composition system can deal with matchmaking of parameters during the composition process. Thus their Linear Logic theorem prover can deal with both the service specification and the semantic annotations.

In the same way as [208] the authors focus on **non-functional parameters** (e.g., quality of services) to model **optimization** criteria. In particular they consider the latter parameters directly in the theorem proving process in order to satisfy the user requirements.
Their method uses **DAML-S** (the former version of OWL-S) as a semantic web service language to describe services and their composition, and then ensures the industrial **applicability** of their approach.

A Semi-Automated Approach

FLC of web services has also been discussed in [188]. Their authors provide a composition tool that supports the end user to **compose** web services in a *semi-automated way.*

In such an approach, services are defined as actions with *restricted expressivity i.e., input and output parameters, without notion of preconditions and effects.* However the result of this approach supports **expressive compositions** such as **sequential, conditional** and **concurrent** compositions.

The end user selects her best web services (depending on her **composability** criteria e.g., **semantic connection** computed at **design time** or **non functional** criteria) for each abstract activity in the composition.
In their approach the possible semantic connections are restricted to the subsumption relation i.e., Exact, PlugIn, Subsume and Disjoint. Unlike [174], they extend the composability with the Subsume matchmaking level. From these different levels of matchmaking type, they introduce a partial order on them such that the rank of the matches are lowered when the distance between the two DL descriptions in the ontology tree increases.

Upon selecting a web service, the services that can produce output that could be fed as the input of the selected service are listed after filtering based on profile descriptions. The restricted list of services are selected on semantic matchmaking retrieved between the latter output and input parameters. The end user can then manually select the service that she wants to fit in at a particular activity. After selecting all the services, the system generates the composite process. The final flow specification to semantically link web services is then created in a semi-automated way. *The semi-automation of the composition process limits* the **flexibility of the model**.

The execution is done by calling each service separately and passing the results between services according to the flow specifications.

In this approach, **optimal** composition of services is computed by considering *local and manual selection of semantic connections.* Indeed, during the composition process, the end-user may only choose the best local semantic web services, depending on its semantic connection with its previous services.

In their approach the end user is required at each service (industrial **applicability** in **DAML-S**) selection step of the composition process (also described in **DAML-S**).

The strength of their system is to enable step-wise composition of services, where the filtered services presented at each step depend on the services chosen at the previous step and their constraints.

A Basic but Efficient Approach

The authors of [105] have addressed in more detail the problem of interleaving web service discovery and composition by applying a **Breadth-first search**.

In more details, they consider a restrictive definition of Web services i.e., *services with at most one input and one output parameters, and no preconditions and effects* (i.e., the simplest case for a service composition). In the same way the **expressivity** of compositions is reduced since they plan to compute *simple workflows wherein only sequence and non determinism are embedded.*

In such a case their breath-first search approach returns a plan of web service restricted to a *sequence of limited web services* hence a *linear workflow of web services*, also defined as a linear and total order of services. With the aim of generating a composite service plan out of existing services, [128] propose a composition path, which is a sequence of operators that compute data, and connectors that provide data transport between operators.

The search for possible services to construct a sequence is based on the following **composability criteria**: **semantic similarity between output and input parameters**. *In the same way as [188], their approach restricts the matchmaking functions to be Exact, PlugIn, Subsume and Disjoint.*

To this end they exploit a **semantic reasoner** to detect the subsumption relationships between functional parameters at **design time**.

The lack of expressivity in service and composition is the main limitation to obtain a flexible model.

In this approach, **optimal** composition of services is computed by means of on the shortest composition i.e., the composition wherein the less number of services are involved.

The industrial **applicability** of their approach is further restricted since they assume *a meta language based on RDF* to describe the services. On contrary the language used to describe the composite service is independent of the suggested approach by [105].

A Graph Theory based Approach

In [15] (also discussed in [14, 210]) the authors present one of the most advanced approach in functional **semantic dependence** based automated web service composition. They introduce a method for automatic composition of semantic web services using **Dynamic Programming** (**flexibility**) coupled with **Graph Theory** as a **composition mechanism**.

In the same way as most of approaches that perform FLC at semantic level, the **expressivity of Web services** is reduced to semantic annotations of *input and output parameters*. Even if their approach support **sequence** and **concurrency** as **composition constructs**, *conditional* compositions cannot hold in the final result (only Web services with determinist output parameters).

In a nutshell, their approach is as follows. From an end user goal, described in terms of both expected output parameters and required input parameters, they aim at computing the best composition of services that match the user goal. In other words inputs and outputs of the composite service should match the user-supplied inputs, and expected outputs, respectively.

Towards this issue they consider semantic web service composition as a directed graph wherein nodes refer to web services. Those nodes are linked by edges referring to a **semantic dependence** (as a **composability**) between an output and an input parameter of two web services. *The matching compatibility is valued by standard matchmaking type i.e., Exact, Subsume, PlugIn, Disjoint* [152].

To this end they exploit a **semantic reasoner** to detect the subsumption relationships between functional parameters at *run time*.

From this graph they suggested an approach to retrieve compositions by navigating through the graph to find the shortest sequences starting from the initial requirements (i.e., end user's input parameters) and go forward by chaining services until they deliver the goal (i.e., user's expected output parameters). The composition terminates when a set of services that matches all expected output parameters is found, or the system fails to generate such a composition of services.

The goal of their approach is to find an optimal composition of services that produces the desired outputs by considering non functional quality of services together with semantic matching between functional parameters. The **optimal composition** is selected on the number of services involved in the final composition. To this end they adapt the Bellman-Ford shortest (optimal)-path dynamic programming algorithm to find the shortest sequence from initial node to the termination node. Their exhaustive search of the optimal composition among all the candidate compositions, making this approach impractical for large scale composition.

They define the **applicability** of their work to services described in **DAML-S** or **SA-WSDL**. The computed composite Web service are **independent** of their approach.

Synthesis

a) Matchmaking based Discovery vs. Matchmaking based FLC:

As observed in the latter models, achieving matchmaking in FLC requires some adaptations of the pure matchmaking based discovery process.

First of all, the matchmaking steps we need to perform in FLC are between output parameters of services and input parameters of other services, and not between same category of parameters such as in pure matchmaking tasks.

Secondly, FLC aims at retrieving a composition as a partial order of services that achieve the service request and support **expressive compositions** such as sequential, conditional and concurrent compositions. On contrary the matchmaking task aims at discovering "the" best advertised service that semantically matches with the requested service. Some issues related to control flow and data flow (semantic dependences) are key in FLC whereas the latter issues do not need to be considered in the matchmaking task.

Finally, unlike the matchmaking based discovery, other parameters such as precondition and effect expressions of services need to be considered in FLC.

b) Advantages and Limitations of Matchmaking based FLC:

In this section we sketch advantages and limitations of the previous models along six main properties:

- **Automation** of the composition process:
  Each model is described according to its **Composition mechanism**, and **Formalism** used to model Web service and and composition results.
- **Expressivity** of Service and their Composition:
  As observed in the latter models, **expressivity on Web service description** and **expressivity on their composition description** is, in many cases, under specified.
  - On the one hand input and output parameters based matchmaking is not enough to compose at functional level since Web services may have same input and output parameters and different functionalities. Moreover such a limitation can affect the final composition result, by for instance, retrieving some compositions with open preconditions or inappropriate effects. Therefore preconditions and effects needs also to be considered in composition process.
  - On the other hand structured composite services prescribe the order in which a collection of activities (here services) take place. They describe how a service is created by composing the basic activities it performs into structures that express the control patterns, data flow, handling of faults and external events, and coordination of message exchanges between service instances. In this work, the expressivity of a composition states the control constructs the composition can handle. We identify the following three groups of control constructs for assembling primitive actions into a complex actions that collectively comprise an applications:
    * Sequential ordering;
    * Conditional compositions;
    * Concurrency, namely: parallel split, synchronization and exclusive choice.

    These constructs seem the minimum set of composition constructs to model expressive compositions of Web services.

- **Composability** criteria used to Relate Some Web Services in a Composition:
  Considering **semantic dependence** as a **composability** criteria rather than *syntactic equivalence* between functional parameters of Web services makes service composition more robust. Indeed output parameters of a service that does not match exactly the input parameters of another service can be considered in Web service composition. Therefore such models are mainly oriented by semantic matchmaking between output and input parameters of services.

  However, computed at **design or run time**, the semantic dependences are, at best, defined as an Exact, PlugIn, Subsume or Disjoint matchmaking between two functional parameters of Web services in the latter models. In case the domain ontology is complex, computing subsumption relationship between functional parameters at run time can be a challenging task.

  Even if matchmaking based approach solves some heterogeneity problem by using semantic description of functional parameters, there are still some limitations. For instance it is still semantically safe to transfer the more specific output to the more general input (i.e., PlugIn matchmaking), but its seems more challenging to achieve a Subsume match type between an output and an input parameter of two services . Indeed such a matchmaking should not be directly applied in a web service composition since the output parameter is not specific and specified enough (e.g., in this case an input parameter requires more description than provided by the output parameter) to be exploited by the input parameter. The input parameter is said to be over specified and the output parameter to be under specified. Therefore the Subsume matchmaking type cannot be directly used in FLC.

  Moreover, as previously studied in section 1.2 there are several proposals such as Abduction [55], Contraction [49], Difference [35, 198] that can be re-used to extend the matchmaking functions and then exploit more different levels of semantic similarities and properties (e.g., applying contraction between services parameters in case their intersection is empty).

- **Flexibility** of the composition model:
  The **flexibility** of the model is important to further facilitate extensions and adaptability to complex systems with standard components of publishing, discovery, execution engine.

- **Optimzation**:
  Besides computing composition of Web services, most of these approaches consider an **optimization** process to compute the *best composition*. Such a process is important to prune the set of candidate compositions that can achieve a same goal.

  On the one hand, services consume resources, such as network bandwidth, and may have a monetary cost associated with their execution. Therefore, depending on a given criterion e.g., *non-functional parameters* (quality of services such as price, execution time, and reliability [208, 13]), *number of services involved in the composition*, the properties quality of the *best* computed composition varies e.g., cheapest (in terms of price), fastest (in terms of time execution).

  On the other hand, the semantic form of web service enables different levels of matchmaking between their functional parameters. Therefore different metrics can be used to optimize service selection as well as the resulting composition. For instance, criteria based on semantic dependence between Web services, could be also considered to value semantics of a composition. However the latter approaches are, unfortunately, only non functional parameters oriented (e.g., quality of service). Even if such an optimization is adapted for web service compositions, it is far from convenient for "semantic" web service compositions.

- Industrial **Applicability** of the approach:
  The composition approaches that support standard proposals such as DAML-S, SWSO, SA-WSDL, WSMO or BPEL4WS (a language appropriate for tasks such as verification) ensure the industrial **applicability** of their approach in real scenario. This property is key to further value and compare approaches e.g., scalability, computation time performance.

The previous six properties of each previous models are summarized in Table 2.1.

2.1.3 AI Planning and Causality Relationships based Composition

In this section we first present some backgrounds on AI planning based composition.

From this definition, we review main related work in *AI planning and causality relationships based composition* i.e., approaches that compute composition by means of relationships between effects and preconditions of web services.

However we do not plan to present an exhaustive list of the methods that meet this AI planning based model such as rule-based Planning [141, 180], conditional planning [103] or planning as model checking [169]. On contrary we mainly focus on the closest approaches of our model i.e., an HTN Planning oriented approach [191], a Situation Calculus oriented approach [139], an estimated regression planning based approach [135], a conformant Planning approach with Incomplete Information [89] and a metric planning based approach [4].

Concepts and Definitions

AI Planning [178], in general, can be regarded as an area of study that is concerned with *automatic generation of plans* that will be able to solve a problem within a particular domain.

Typically, a plan consists of sequence of actions, such that given an initial state or a condition, a planner will suitably select a set of actions which, when executed according to the generated plan, will satisfy certain goal conditions.

Web services are an interesting domain for planning research, since they fit the classical model of discrete actions (i.e., simple procedure calls), but obviously require eliminating closed-world assumptions since complete information cannot be ensured. Moreover modelling multi branching plans in service composition is another main requirement.

In the context of web services, a planning domain can be represented by a six-tuple $\langle W, S, A, \Gamma, s_0, G \rangle$, where

- W is the set of available web services,
- S is the set of all possible states of these services (i.e., the world),
- A is the set of actions/functions provided by the services that the planner can perform in attempting to change the state from one to another in the world,
- $\Gamma \subseteq S \times A \times S$ is the set of state transitions which denote the preconditions and effects for execution of each action,
- and finally $s_0 \in S$ and $G \in S$ are the initial and goal states, respectively, specified in the requirement of the web service requesters to indicate that the plan initiates its execution starting from state s_0 and terminates at state G.

References			[174]	[188]	[105]	[15]
Automation	Formalism		Linear Logic	Independent	Similarity with HTN	Graph Theory
	Composition/Search Mechanism		Theorem Proving	No Automation	Breadth-first Search	Dynamic Programming
Expressivity	Service Expressivity (R_1)	Input	✓	✓	restricted to 1	✓
		Precondition	✗	✗	✗	✗
		Output	✓	✓	restricted to 1	✓
		Effect	✗	✗	✗	✗
	Composition Expressivity (R_2)	Sequence	✓	✓	✓	✓
		Non Determinism	✗	✓	✓	✗
		Concurrency	✓	✓	✗	✓
Composability	Semantic Dependences	Matchmaking Function (R_4)	Exact, PlugIn Disjoint	Exact, PlugIn Subsume, Disjoint	Exact, PlugIn Subsume, Disjoint	Exact, PlugIn Subsume, Disjoint
		DL Reasoning (R_8)	Design Time	Design Time	Design Time	Run Time
Flexibility (R_6)			✓	✗	✗	✓
Optimization (R_7)			Non functional Parameter (e.g., QoS)	Local and Manual on semantic connections	Number of services	Number of services
Applicability	Web Services		DAML-S	DAML-S	RDF	DAML-S, SA-WSDL
	Service Composition		DAML-S	DAML-S	Independent	Independent

Table 2.1: Some Matchmaking and Semantic Dependences based FLC Approaches. Legend: ✗ = not supported, ✓= fully supported.

Given this domain, many approaches have been proposed by applying a variety of planning techniques that will automatically generate a plan (i.e., a composition) for realizing the goal requirements.

Most of these systems are directed by *causality relationships* between effects and preconditions of services (see Section 2.1.1) based AI planning where only preconditions and effects are considered to represent web services. For instance such a subset is modelled by the Successor State Axioms in Golog [122].

This is the main conceptual difference with the *semantic dependences* based FLC.

In the next section we will focus on related work of web service composition that performs FLC, mainly, by means of *causality relationships* between effects and preconditions of services. The following four works have been studied in details for these main reasons.

A Hierarchial Task Network based Approach

In [191] (also discussed in [205]) the authors describe an hand-coded planning systems, **Simple Hierarchical Ordered Planner 2 (SHOP2)** to support their **composition mechanism**. Hand-coded planners such as SHOP2 are domain-independent planning system, which use domain-specific control knowledge to help them plan effectively.

SHOP2 is based on **hierarchical task network (HTN) planning**, which is an AI planning methodology that creates plan by task decomposition.

Since authors of [191] interface the composition problem to an HTN planning approach, Web services with input, output parameters, preconditions and effects can be considered. However their **expressivity** is reduced since they consider *Web services to be either information-providing or world-altering Web services.*

Even if SHOP2 [191] supports conditional effects in composition, the returned plan is a *sequence* of selected and executed service that achieve the goal. They also do not address methods to retrieve the whole set of feasible solutions.

Indeed there is no mechanism to handle the control constructs related to concurrency. Such a limitation in SHOP2 imposes a serious limitation on the usefulness of this methodology.

At the moment this is resolved by enumerating every possible flow in the process using conditional expressions in the method descriptions. This increases the complexity of search space, and planning.

Planning progresses as a recursive application of the methods to **decompose tasks into smaller and smaller subtasks** (**composability criteria**[1]), until the primitive tasks, which can be performed directly using the planning operators, are reached.

In the case where the plan later turns out to be infeasible, SHOP2 will backtrack and try other applicable methods. Note that **semantic dependences between input and output parameters are also considered to ensure data flow in Web service composition** (in particular the restricted list: Exact, PlugIn, Disjoint).

One difference between SHOP2 and most other HTN planning systems is that SHOP2 plans for tasks in the same order that they will later be executed. Planning for tasks in the order they will be performed makes it possible to know the current state of the world at each step in

[1] Here, the notion of causality relationship between effects and preconditions of Web services is more abstract, and defined by a process of static decomposition of tasks in smaller subtasks.

the planning process, which makes it possible for SHOP2's preconditions evaluation mechanism to incorporate significant inference and reasoning power, including the ability to call external programs.

This allows SHOP2 to integrate planning with external information sources as in the Web environment. In order to do planning in a given planning domain, SHOP2 needs to be given the knowledge about that domain. SHOP2's knowledge base contains operators and methods. Each operator is a description of what needs to be done to accomplish some primitive task, and each method tells how to decompose some compound task into partially ordered subtasks.

The main disadvantage of this approach is that whilst hand-coded search does help them plan effectively, it creates a significant overhead. Consequently it requires expertise in both the domain and specifics of the planner.

Moreover the model together with its composition result are mainly dependent on the knowledge base of the domain i.e., a detailed description of compound tasks decomposed into partially ordered primitive tasks. Such a constraint puts limitations on level of automation of composition, and limits the **flexibility** of the approach since decomposition of web services required to be known at composition time (i.e., **static decomposition tasks process**).

In SHOP2 the goal cannot be stated declaratively. SHOP2 has to know in advance which method it should call. Consequently the planner fails if asked to solve a completely new, unknown problem for which no method definition exists.

The authors work with these limitations since SHOP2 operate as an online (i.e., run time) planner. In other words during each service selection step, the service is executed in the real world. Indeed, to enable information providing from web services at planning time (i.e., composition time), they require that services to be either exclusively information-providing or exclusively world-altering.

Even if [191] provide an approach to achieve Web service composition, *they do not address the problem of selecting* the **optimal composition** among a set of candidate compositions.

Since the concept of task decomposition in HTN is very similar to the concept of process decomposition in OWL-S, the SHOP2 approach is quite **adapted and applicable** for **OWL-S web services**. In this direction, they encode a composition problem as a SHOP2 planning problem, so SHOP2 can be used with OWL-S web service descriptions to automatically generate a composition of services.

A Golog based Approach

The authors of [139] (also discussed in [140]) encode composite services in **Golog** [122], **a high-level logic programming language built on top of the situation calculus**. In such an approach, the **composition mechanism** follows **Breadth first planning** approach.

In their Situation Calculus-based framework **a service is specified as Golog** [122] **procedures** and seen by end-users as **an atomic action, thus presenting an input/output behaviour**. Moreover they restrict their composition model by assuming a complete independence between information providing and world-altering actions (in the same way as [191]).

Such an assumption is very restrictive in the open world of web services . A situation tree (i.e., a kind of process flow in the theory of Situation Calculus) is associated with each atomic action. From these atomic actions, the results of their approach is a simple linear *sequences* of web services (viewed as a total order).

Unfortunately *alternative and conditional compositions cannot be computed* at composition time since [139] compute only online compositions and then execute services at composition time.

Since concurrency of services in a composition is not handled in such an approach, [165] suggest to use an extended version of Golog i.e., ConGolog [78] to overcome the problem of concurrent execution in semantic Web services.

In their approach, the **composability criterion** is defined by *causality relationships* between effects and preconditions of some services.

The composability criteria of the composition problem are encoded by the *Successor State Axioms* of the Situation Calculus-based framework, which also provide a solution to the frame problem[2] [176].

Contrary to the semantic matchmaking based composition approaches, semantic dependences between input and output parameters are not considered. This restricts the data flow to be purely *syntactic*.

To support information providing services combined with world altering services, they propose a middle-ground Golog interpreter that i) interleaves offline (i.e., design time) and online (i.e., run time) execution, and ii) operates under an assumption of *reasonable persistence* of certain information.

The reasonable persistence of information assumption prevents the planner to change (i.e. simulate the changes) the information provided from external sources.

In the open and dynamic world of web services, there are many scenarios (e.g., in Telecommunication domain) where the reasonable persistence of information, as suggested by [139] (and re-used by [191]), cannot hold. Indeed information has a limited temporal extent associated with it, which may affect composition. Therefore we cannot make such an assumption in our composition issue.

A ConGolog interpreter is augmented with online execution of information-providing services with offline simulation of world altering services. The approach used to combine online execution of sensing actions with offline simulation of world altering services is very similar to the [191]'s approach in spirit.

One advantage of using situation calculus as the underlying logical framework is the additional expressivity and the ability to do arbitrary reasoning about first-order theories.

The **flexibility** of their approach is limited by this *on-line execution*. For instance, Web service composition conditioned by output parameters of services cannot be supported in this approach.

The authors of [139] have considered programs that are **generic, customizable by end user (e.g., through preferences) and usable** in the context of the Web. In this direction, their adaptation and extension of the Golog language support a **selection criterion** (i.e., **preference**) to prune the enormous search space, but no criteria are suggested to distinguished similar services and compositions.

[2]The frame problem is one of the central theoretical issues of Artificial Intelligence. Roughly speaking, to most AI researchers, the frame problem is the challenge of representing the effects of action in logic without having to represent explicitly a large number of intuitively obvious non-effects [178].

The **applicability** of their approach is defined by a relevant subset [144] of DAML-S in terms of the situation calculus. Atomic service descriptions, preconditions and effects in DAML-S are mapped to situation calculus constructs. On contrary the composition result is neither a service (in the sense that it cannot be re-used by another client) and nor mapped to a standard composition language since the composition is directly executed step by step (i.e., on-line execution).

A Metric Planning based approach

The authors of [4] (also discussed in [5]) study Web service composition with **SEMAPLAN** by combining **sem**antic matching and an AI **plan**ning algorithm. In such an approach, the **composition mechanism** follows **Forward search** (aka progression based) approach on a metric planning problem. Similarly to *matchmaking and semantic dependences based composition* approaches they compute overall semantic similarity scores (metric based) between services. In this direction they address *planning with partial semantic matchmaking* (i.e., an approach in between AI planning and Matchmaking based approaches).

In their approach the concept of input is very close to the concept of precondition. In the same way the notion of output is very close to the notion of effect. In particular **semantic annotation on their input and output parameters** are mainly considered to describe Web services of the domain.

The expressivity of the computed composition is limited to *sequential* composition, and unfortunately *no process to model concurrent and conditional compositions is described.*

The **composability criterion** is defined by *causality relationships* between parameters of services. Due to the close definition of input, precondition and output, effect, *causality relationships* is achieved by **semantic relationships** between the latter parameters. In particular they focus on a restricted list of matchmaking functions (i.e., **Exact, PlugIn, Subsume and Disjoint**) to elaborate relationships between services.

The **flexibility** of their approach is limited by the expressivity of both the service description and the composition constructs they support.

The authors of [4] present a **ranking module** to select compositions depending on different criteria. For instance the optimal composition can be selected by means of its **plan length** (i.e., the number of services in the plan), or **local selection of semantic relationships between services** (i.e., a semantic cost of the plan).

Their method uses **WSDL-S** (the former version of SA-WSDL) as a semantic web service language to describe services they plan to compose. The computed composition can be supported by any composition language. This ensures the industrial **applicability** of their approach.

An Estimated Regression Planning based Approach

The author of [135] investigates applicability of **estimated regression planners** for generating compositions of web services. Unlike [191] and [139] they focus on their conditional form. The estimated regression planners use a backward analysis of difficulty of a goal to guide a forward search through situation space. By extending the **Unpop planner** [134] the author created the **Optop mechanism** i.e., an Opt-based total-order planner.

In [135], the author introduces a new type of knowledge, called value of an action, essentially representing certain information that is created or learned as a consequence of executing a particular action.

The main intention of introducing this extension was to have the ability to capture the information and the content of messages that are exchanged between the services. Such information persists and is not treated as a truth literal.

From **Web service construction and expressivity** perspective, such a feature is required to distinguish the information transformation and the state change produced by the execution of the service. The information, which is presented by the **input and output parameters** are thought to be reusable, thus the data values can be reused for the execution of multiple services. Contrarily, the states of the world are changed by the service execution (**through preconditions and effects**). The change is interpreted as that the old states are consumed and the new states are produced.

Contrary to [139], the composition result of [135] supports **more expressive compositions**. Indeed the computed composition can be modelled by *sequence and non determinism* whereas [135] cannot model such conditional compositions. However composition constructs such as *concurrency* is not handled.

In the Optop mechanism, a state of the planner is a situation, which is essentially the current partial plan. Optop works with classical-planning goals; thus, it checks whether the current situation satisfies the conjunction of the goal literals given to the planner as input. During its search, Optop computes a regression match graph as described in [135], which essentially provides information about how to reach a goal state from the current situation.

The **composability criteria** used by the regression search is based on *causality relationships* between effects and preconditions of services. The planner returns the successor situations that arises from applying the actions specified by the latter graph in the current situation.

In the same way as [139], they compile composition under the assumption of exact matches between functional parameters, hence no way to value semantic dependences between web services at semantic level.

The main motivation for this work is to relax the assumption of complete knowledge required by classical planners, and to formalize what they do not know and how they could find out more about the world.

The author of [135] also points out the necessity for planners to support synthesis of branching (i.e., conditional plans) and looping plans. These ensure the **flexibility** of their approach.

Even if the Optop planner is quite appropriate not only to compute conditional compositions of services but also to operate in the open world of Web (by introducing the value of an action), **no optimization criteria** is given to select an *optimal* composition among several candidates.

In such an approach Web services and their composition are represented as PDDL actions, and unfortunately not by means of a standard proposal such as SA-WSDL, WSMO or OWL-S. This restricts the industrial **applicability** of the approach. However some prototype tools can be tested and used to translate PDDL to DAML [3].

[3] http://www.cs.yale.edu/homes/dvm/daml/pddl_daml_translator1.html

A Conformant Planning based Approach

In the same way as [135], the authors of [89] focus on the automated generation of conditional compositions of Web services. To this end, they investigate applicability of the **(CFF planner i.e., Conformant Fast Forward planner)** [90] i.e., an off-the-shelf planning **mechanism** that operate under uncertainty the whole under certain additional restrictions.

In their approach they consider the case where there is no observability, i.e., conformant planning. At best, observability in their context is partial. This means that "observing" during service execution may involve requesting additional information from the user, or even crossing enterprise boundaries.

Besides **semantic annotation on their input and output parameters**, [89] consider actions with **conditional effects**, which would normally allow several conditional effects for each action, as well as **forced preconditions** (which must be known to hold, for the action to be executable).

Contrary to [191], [139] and [135] they consider more **expressive composition** of Web services. Indeed their model handle constructs that support **sequential, concurrent and conditional** compositions.

In [89], the high complexity of planning in web service composition motivates the search for interesting special cases. In this directions, the authors formalize a *special case* of Web service composition, termed "forward effect" where the semantics is an important feature to consider.

Indeed, besides ensuring the **causality relationships between effects and preconditions of services**, they model **composability** between Web services as **semantic connection** (so, called *partial matches in their work*) by "forward effect".

In the same way as [191], they first argue that exact match cannot be sufficient in Web service composition based AI planning. Indeed some compositions may require so-called partial matches, where a service may provide only part of parameters required by another service. Therefore their core observation is based on a notion of compatible actions (i.e., services) i.e., actions that can achieve a partial match. Uncertainty is then based on these partial matches between Web services. *However such match levels are very restrictive to perform composition of services.*

From this the authors suggest an innovative approach i.e., considering "limited" partial match (i.e., a non empty intersection of output parameters to achieve an input parameter of a service) together with conditional compositions.

Even if [89] suggest partial matches to overcome web service composition, they do not study in detail the limitation of their matchmaking types e.g., if the partial match of n output parameters subsume an input parameter of another service, matchmaking of those parameters is not appropriate to perform a composition.

Moreover no detail about the computation of partial matches is presented in [89]. No information states if DL reasoning is pre-computed or achieved at composition time.

The high **flexibility** of their approach is mainly due to i) the expressivity they consider to model Web services and their composition and ii) their two different levels of composability criteria.

No optimization process is provided to reduce the set of possible Web service composition candidates.

Their composition approach is **applicable** with Web services described in WSMO (through a PDDL translation), and provide composition of Web services in an independent language that can be re-used once composition is achieved.

Other AI Planning based Approaches

Note that latter forms of composition model are tightly related to *classical planning* in AI, and have been adopted by many other works e.g., [9]. These, although different in the kind of goals and initial conditions, are all based on the idea of sequential compositions of the available services, which are considered as black boxes.

Advantages and Limitations of AI Planning based FLC

In the same way as Section 2.1.2 we sketched advantages and limitations of the previous models along the following six key properties:

- **Automation** of the composition process.
  Each model is described according its **Composition mechanism**, and **Formalism** used to model Web service and composition.
- **Expressivity** of Service and their Composition:
  As observed in the latter models, **expressivity on Web service description** and **expressivity on their composition description** is, in most of cases, under specified.
  - On the one hand preconditions and effects of Web services are not enough to compose at functional level since such a level of composition requires also the knowledge of the data flow. Therefore input and output parameters as semantic annotations of parameters need also to be considered in composition process.
  - On the other hand (and in the same way as approaches presented in Section 2.1.2), sequential, conditional and concurrent compositions seem the minimum set of basic compositions we need to consider. Contrary to the *AI planning and causality relationships based composition* approaches, concurrency is addressed in most of *Matchmaking and semantic dependences based composition* approaches. Such expressive level of composition is often disregarded in AI planning based FLC.
- **Composability** Criteria Used to Relate Some Web Services in a Composition:
  Contrary to the *matchmaking* based composition approaches, *causality relationships* between preconditions and effects of Web services are considered as the main important feature to ensure composability between Web services in AI planning based composition approaches. In particular such approaches enable us to compute compositions wherein any preconditions could be satisfied by an effect of another service. However Web service composition could be achieve by coupling the **causality relationships** with the **semantic dependence** between input and output parameters to satisfy both i) flexibility and ii) data flow.

  In the same way as approaches presented in Section 2.1.2, approaches that partially consider semantic matchmaking between functional parameters do not study in detail the limitations and properties of the matchmaking types they use to compose services.
- **Flexibility** of the Composition Model:
  The **flexibility** of the model is important to further facilitate extensions and adaptability to complex systems with standard components of publishing, discovery, execution engine.

- **Optimzation**:
  Besides computing composition of Web services, some approaches consider an **optimization** process to compute the *best composition*. Such a process is important to prune the set of candidate compositions that can achieve a same goal.

- Industrial **Applicability** of the approach:
  The composition approaches that support standard proposals such as DAML-S, SWSO, WSMO, or BPEL4WS ensure the **applicability** of their approach in industrial scenario. According to the definition of SA-WSDL, preconditions and effects of SA-WSDL services will not be considered. This remains an important criterion to value this level of composition approaches.

The six properties of each previous models are summarized in Table 2.2.

2.1.4 Synthesis

Although the previous approaches, combined with semantic reasoning and/or AI planning, offer practical approaches to perform *automation* of Web services composition, both latter approaches encounter some limitations and require some extensions.

On the one hand considering preconditions and effects is one open issue for the *Matchmaking and semantic dependences based* systems whereas designing correct, complex compositions (through, for instance, concurrent, conditional compmosition) and data flow are two main requirements for these systems.

On the other hand potentially enormous search space and the difficulty in fully and accurately representing real-world problems are two key challenges for the *AI planning and causality relationships based* systems whereas *causality relationships*, and more generally *causal laws* are main requirements for these systems.

Finally, semantic dependences between Web services is reduced to the valuation of standard matchmaking function (e.g., PlugIn, Subsume, Disjoint) in both approaches.

In this section we posed six specific technical requirements, summarized in Table 2.3, that need to be met by AI planning and matchmaking based systems to make automatic web service composition at functional level a real success.

- **Automation** ($R^{Composition}_{Automation}$)
  - the **Composition Mechanism** used to achieve automation of composition;
  - the **Formalism** used to describe the composition problem.
- **Expressivity** ($R_{Expressivity}$)
  - of service ($R^{Service}_{Expressivity}$);
  - of their composition ($R^{Composition}_{Expressivity}$).
- **Composability** ($R_{Composability}$) criteria pertaining automatic composition of services, described in terms of input, output parameters and preconditions, effects. Such criteria are used to relate some Web services in a composition:

References			[139]	[191]	[89]	[135]	[4]
Automation	Formalism		Situation Calculus	Hierarchical Task Network	Conformant Planning (under uncertainty)	Estimated Regression Planning	Independent
	Composition/Search Mechanism		Golog + *World Simplest Breath First Plan-ner* (wsbfp) [177]	SHOP2 [145]	Conformant Fast Forward planner (CFF) [90] (conditional planner)	Extension of Unpop [134] i.e., Optop (conditional planner)	Forward Search (Semaplan)
Expressivity	Service Expressivity	Input	✓	✓	✓	✓	✓
		Precondition	✓	✓	✓	✓	✓(close to input)
		Output	without Effect	without Effect	✓	✓	✓
		Effect	without Output	without Output	✓	✓	✓(close to output)
		Category	Information-Providing or World-Altering	Information-Providing or World-Altering	Any	Any	Information-Providing
		Assumption	Persistent Information	Persistent Information	✗	✗	✗
	Composition Expressivity (R_2)	Sequence	✓	✓	✓	✓	✓
		Non Determinism	✗	✗	✓	✓	✗
		Concurrency	✗ (✓in [165])	✗	✓	✗	✗
Composability	Semantic Dependences	Matchmaking Function (R_4)	Exact	Exact, PlugIn, Disjoint	Partial Matches	Exact	Exact, PlugIn, Subsume, Disjoint
		DL Reasoning (R_8)	✗	Design Time	?	✗	Design Time
	Causality Relationships (Part of Causal Laws)		Successor State Axioms	Task/ Method Decomposition	Forward Effect	Causality Relationships between Effect and precondition	Causality Relationships (close to semantic dependence)
Flexibility (R_6)			Limited by the Online Execution	Limited by the decomposition tasks process	✓	✓	✗
Optimization			User Preference	✗	✗	✗	Number of services, (Local) Semantics
Applicability	Web Services		DAML-S	OWL-S	WSMO	PDDL Actions	WSDL-S
	Service Composition		✗	OWL-S	Independent	PDDL AND/OR Tree	Independent

Table 2.2: Some AI Planning and Causality Relationships based FLC. Legend: ✗ = not supported, ✓= fully supported, ? = No Information.

  - semantic dependences ($R^{Semantic}_{Composability}$);
  - causal laws ($R^{Causal}_{Composability}$).
- **Flexibility** ($R_{Flexibility}$) of the composition model.
- **Optimization** ($R_{Optimization}$).
- Industrial **Applicability** ($R_{Applicability}$) of the approach
  - with standard service languages ($R^{Service}_{Applicability}$);
  - with standard composition languages ($R^{Composition}_{Applicability}$).

2.2 Process Level Composition

In this section, we first describe *process level composition* and focus on their *behavioural* features. Then we will briefly review some related works in the area of this level composition, that can be applied with our FLC approach to perform an end-to-end composition of both *functional* and *process* based Web services.

Such a level of composition is not the scope of this Ph.D work, however the reader can be interested in selecting a *process level composer* to combine it with, for instance, our *functional level* composer described in Chapters 3, 4, and 5. This is the main motivation of this section.

2.2.1 Description

Process Level Compositions (henceforth PLC), also known as process-oriented or control flow driven compositions [25, 41, 144, 170, 201] consider compositions of web services which are described on a process view.

As highlighted in Section 1.1.3 *process level* based Web services are viewed no more as capabilities, but as stateful and asynchronous finite state machines, which establish multi-phase protocols to define agreements and to exchange data, and whose final outcomes may be not fully predictable i.e., non deterministic. As a consequence, also the generated executable code responsible of the composition is a complex behaviour, since all possible contingencies occurring in the interaction with the Web services have to be considered.

Contrary to FLC, process level composition (henceforth PLC) covers a finer-grained phase of composition. Since web services are usually described by their internal protocol so called behaviour, PLC aims at describing the following system:

> *PLC aims at describing a system interacting with all protocols involved in the composition.*

The Figure 2.3 illustrates an *ideal process level composition* of Web services that cannot be reduced to an atomic step, but requires instead a sequence of operations, including e.g., authentication, submission of a specific request, negotiation of an offer, acceptance (or refusal) of the offer, and achieving another internal action.

Such a description of web services can be provided, for instance, by means of a subclass of the *Service Model Process* in OWL-S. The same entity can be described by the *Service Choreography and Orchestration Model* in WSMO, as an activity flow or an interaction pattern.

Requirement R_i	Details	Description
$R^{Composition}_{Automation}$	Formalism	Formalism used to model the Web services Composition problem.
	Composition Mechanism	Technique achieved to compute compositions of Web services.
$R_{Expressivity}$	Web Service $R^{Service}_{Expressivity}$	Level of description used to define Web services. To this end, four main parameters are conceivable i.e., **Inputs**, **Outputs**, **Preconditions** and **Effects**. Web services can fall into three different **categories** i.e., **Information-providing**, **World-altering services or both.**
	Composition $R^{Composition}_{Expressivity}$	Level of Web service composition expressivity. This can be measured by the set of **control constructs** supported by the composition approach.
$R_{Composability}$	Semantic Dependences $R^{Semantic}_{Composability}$	Level of **matchmaking functions** used to value semantic dependences between Web services parameters. This matchmaking task, valued by a step of Description Reasoning can be achieved at **design** or **run time**.
	Causal Laws $R^{Causal}_{Composability}$	Level of **Causal Laws**: *Causality relationships* between effects and preconditions of services and *complex relationships* between services (e.g., relationships that link an input parameter of a service to an output parameter of another service under some conditions).
$R_{Flexibility}$		Possibility to further **extensions** and **adaptability** to complex systems
$R_{Optimization}$		Criterion used to **select a composition** among a set of candidates that achieve the same goal.
$R_{Applicability}$	Web Services $R^{Service}_{Applicability}$	**Standards** to which the **service** can apply to (e.g., OWL-S, WSMO, SA-WSDL, SWSO).
	Composition $R^{Composition}_{Applicability}$	**Standards** to which the **composition** result can apply to (e.g., OWL-S, WSMO, BPEL4WS, SWSO).

Table 2.3: Table of Requirements for Functional Level Composition.

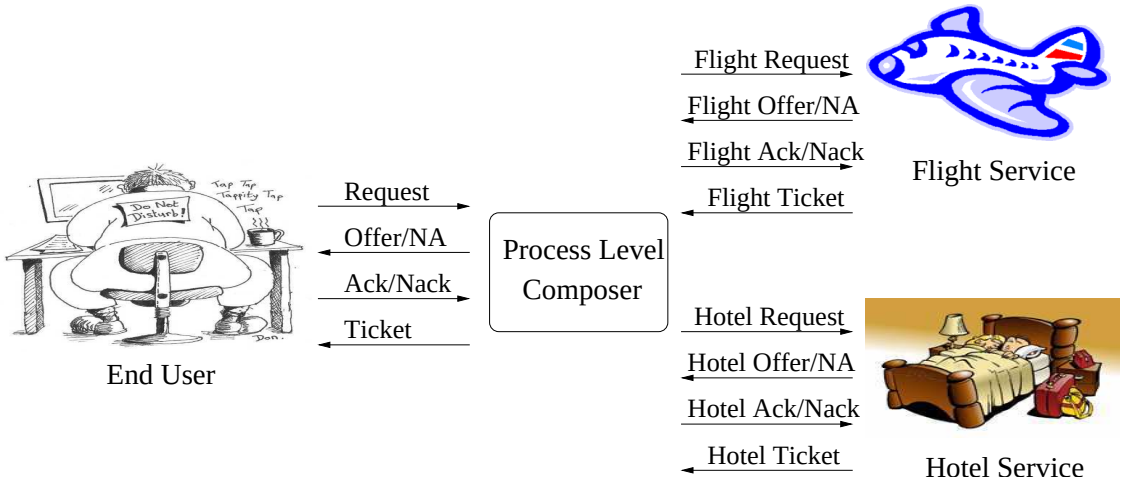


Figure 2.3: An Ideal Process Level Composition of two Web Services (A Virtual Travel Agency).

2.2.2 Some Reference Models

Since our main contribution is related to FLC, we do not focus deeply on the process level. However we briefly describe the three main PLC approaches that can be combined with our FLC approach to achieve an end-to-end composition.

These models are depending on three different entities to achieve composition, ordered by their level of abstraction:

- *message, conversation level*;
- *behavioural level*;
- *abstract activities* i.e., internal actions, message exchanged.

Composition at Behavioural Level

In [167] and [170] the studied composition of services is based on global goals specifying *a complex behaviour*. Such goal are expressed in a request language developed for planning under uncertainty. Moreover they concern the client and all the component services.

From this specification, they present a composition technique that takes as input i) a set of partially specified services, including one representing the client behaviour, modelled as nondeterministic finite state machines, and ii) a global goal expressed as a branching temporal formula. Finally they return a plan that specifies how to coordinate the execution of various services in order to realize the global goal.

The plan can then be encoded in standard coordination languages such as BPEL4WS or OWL-S, and executed by orchestration engines. Note that the composition is mainly concerned with the *global behaviour* of the system in which some client desired executions may happen not to be fulfilled.

Composition at Message Level

a) A First Approach:

In [95], a formal framework is defined for composing web services from behavioural descriptions given in terms of automata. The possible *exchanged messages* are the key component of this approach.

The problem addressed in this work is at the process level, since it considers services that can perform more complex interactions than a simple invoke-response operation. However, the composition problem solved by [95] is different from the problem considered by [167] and [170]. Indeed, in the approach [95], the composition problem is seen as the problem of coordinating the executions of a given set of available services by means of *exchanged messages*, and not as the problem ([167] and [170]) of generating a new composite web service that interacts with the available ones.

Solutions to the former problem can be used to deduce restrictions on an existing (composition automaton representing the) composed service, but not to generate executable code for implementing the composition.

b) A Second Approach known as the Mealy Model:

In [41] a framework for modelling and analyzing the global behaviour of service compositions is presented. Services exchange messages according to a predefined communication topology, expressed as a set of channels among services: a sequence of exchanged messages (as seen by an external virtual watcher) is referred to as conversation.

In this framework properties of conversations are studied in order to characterize the behaviour of services, modelled as Mealy machines. This model is well known as the Mealy model. In such a framework, the composition problem takes in input i) a desired global behaviour (i.e. the set of all possible desired conversations) specified as a Linear Temporal Logic formula, and ii) a composition infrastructure, that is a set of channels, a set of (name of) services and a set of messages.

The output of the synthesis is the specification of the Mealy machines of the services such that their conversations are compliant with the Linear Temporal Logic specification.

Composition at Abstract Activity Level

The work of [26, 27], also known as the *Roman model*, focuses on *activity-based* finite state automata. In this approach, states of the latter automata are *abstract activities* i.e., internal actions or message exchanged as well.

In the same way as [167, 170] they elaborate a composition of available behaviours of Web services in order to obtain a target behaviour. Solutions to this problem can be used to generate executable code for implementing the composition.

This is the main conceptual common view with the work described in [26, 27]. The behaviours are modelled with finite state machine and they describe the composition domain as a determin-

istic PDDL[4] formula wherein target and available behaviours are encoded.

In [26, 27] the behaviour of the services is given, and the synthesis phase reuses them trying to assemble them in order to provide the desired behaviour. The client specification is based on a branching time-semantics: composition focuses on a tree-based structure, where each node denotes a choice point on what to do next.

Reference Models	[95, 41]	[26, 27]	[167, 170]
Abstraction Level (entities used (for composing)	Interaction (Message, Conversation)	Abstract Activities (internal actions, exchanged message)	Behaviour
Composition Goal	Coordinating Execution	Generation of a new Behaviour	
Service Formalism	Mealy Machine	Finite State Machine	
Limitation	Its Abstraction Level	Bounded Number of Services	Independent Services

Table 2.4: Some Reference Process Level Composition. Legend: ✗ = not supported, * = level of expressivity.

Process Level Composition and Other Approaches

For further information and models on *process level composition*, the reader can investigate in the following relevant works:

- Composition at Message Level: [95, 41, 74, 73, 68, 69, 31, 162].
- Composition at Behavioural Level: [167, 170].
- Composition at Abstract Activity Level: [26, 27, 144, 7].

Note that most methods compare composition of web service as a control or synthesis of behaviour (automata) product e.g., product machine [74, 73], product of two interface automata [7], Cartesian product [68, 69, 31], parallel composition of state transition system (extended version of a finite state machine) [162], or their synchronized product [170].

2.2.3 Synthesis

As introduced in this section, the choice of a PLC approach to perform a combined composition with FLC, is depending on the *abstraction level* we plan to support i.e., message, conversation, behaviour, actions (operations).

However the finer this level the better for a process level composition combined with FLC. Indeed FLC operates at the most abstract level i.e. *service level*, and then a combination with message based PLC since more appropriate to achieve an end-to-end composition.

[4]PDDL (Planning Domain Definition Language) [76] is an action-centred language, inspired by the well-known STRIPS formulations of planning problems. At its core is a simple standardisation of the syntax for expressing this familiar semantics of actions, using pre- and post-conditions to describe the applicability and effects of actions.

2.3 Combining Functional and Process Level of Composition

PLC is significantly more complex than FLC, since it is a general form of program synthesis. Indeed a product of web services, as a product of state transition systems, required to be computed in many approaches such as [74, 69, 130], hence a space consuming approach.

Towards this issue it is crucial to run PLC over a restricted number of components services. FLC can be used in this direction by pruning useless discovered services. Indeed the partial order in which FLC sequentializes the relevant services is a very useful indication to guide PLC in searching for ways the protocols (behaviour-based description) of service should interact.

Roughly speaking, once FLC has extracted a set of services whose capabilities can be combined together to obtain the capability of the top-level requirement, we can finally try and combine their actual, stateful implementations.

For instance the behaviour-based description of Web service together with their semantic description and the partial order of services identified by FLC are used as inputs of the [130]'s model to perform PLC. The result of this work is a new and executable protocol that interacts with the existing services to achieve the top-level requirement.

In an alternative approach [30] perform web service composition by applying and interleaving discovery, FLC and then PLC. Two state-of-the-art FLC [89] and PLC [167] approaches are coupled with a naive discovery algorithm to perform an end to end composition.

In another direction [111] consider composition of a subset of web services i.e., stateful and independent web services. Such a composition is achieved by a pre-processing of FLC, followed by a subset of PLC. Since they consider independence of web services, interleaving and synchronization of actions from different web services are not required and then are not considered in this subset of PLC.

It is obvious that a combination of FLC and PLC (or any subset) is not always required e.g., in scenarios where web services do not expose their behaviours (i.e., stateless web services), composition can be achieved by leaving out the PLC step and adopting only FLC.

Since FLC is required i) to compose stateless services, and also to reduce the complexity computation of PLC or β-composition, we suggest to investigate this important level of composition in more details.

Before such an investigation, we briefly review two main models to represent web service composition i.e., Orchestration and Choreography.

2.4 Modeling Web Service Composition

Currently, there are competing initiatives for modelling business process as composite services, which aim to define and manage business process activities and business interaction protocols comprising collaborating services. The terms *Orchestration* and *Choreography* have been widely used to describe business interaction protocols comprising collaborating services. In the following we shall briefly review these higher-level specifications for modelling web service composition.

However, note that the sharp distinction between orchestration and choreography is rather artificial [159] and it is widely believed that they should both coalesce in the confines of a single

language and environment.

2.4.1 Orchestration

Orchestration describes how services can interact with each other at the message level, including the business logic and execution order of the interactions from the perspective and under control of a single endpoint. Orchestration refers to an executable business process that may result in a long-lived, transactional, multi-step process model. With orchestration, the business process interactions are always controlled from the (private) perspective of one of the business parties involved in the process.

It is obvious that modelling the business process of *functional* and *process* level compositions can be both achieved by orchestration. On the one hand composition constructs required by FLC (e.g., basic constructs related to sequence, concurrent execution, non determinism) are largely embedded in orchestration. On the other hand the message level based orchestration is really appropriate for PLC.

Currently, orchestration is targeted by a family of XML-based process standard definition languages most representative of which is the Business Process Execution Language for Web Services (BPEL4WS) [11].

2.4.2 Choreography

Choreography is typically associated with the public (globally visible) message exchanges, rules of interaction and agreements that occur between multiple business process endpoints, rather than a specific business process that is executed by a single party. Choreography is more collaborative in nature than orchestration. It is described from the perspectives of all parts (common view), and defines the complementary observable behaviour between participants in business process collaboration. Choreography tracks the sequence of messages that may involve multiple parties and multiple sources, including customers, suppliers, and partners, where each party involved in the process describes the part they play in the interaction and no party owns the conversation.

Modelling the business process of *process level composition* can be achieved by choreography. Indeed the choreography of services is mainly interesting in exchanged messages and behavioural activities in a composition, which ease the PLC modelling.

On contrary, the choreography based approach to model FLC is not appropriate since this level of composition focuses on atomic interactions.

Service choreography is targeted by Web Services Choreography Description Language (WS-CDL) [97], which specifies the common observable behaviour of all participants engaged in business collaboration.

2.5 Conclusion

Some of the most notable research challenges for the near future for the services composition layer include composability analysis for substitution, compatibility, and conformance for dynamic and adaptive processes, adaptive service compositions, autonomic composition of services, and QoS-aware service compositions.

This chapter has aimed to give an overview of recent progress in automated Web services composition techniques. To this end, we introduced *Process Level Composition* and focus more specially on *functional level composition*. The introduction of the *Process Level Composition* has been motivated by its complementary feature with the *Functional Level*. Main methods that achieve both levels of composition have been studied. In particular we focus our attention on *Matchmaking and semantic dependences based composition* and *AI planning and causality relationships based composition* to perform functional composition of Web services. From the study of functional level of composition we suggested six specific technical and meta requirements (i.e., $R^{Composition}_{Automation}$, $R_{Expressivity}$, $R_{Applicability}$, $R_{Composability}$, $R_{Flexibility}$ and $R_{Optimization}$) that need to be met by AI planning and matchmaking based systems to make automatic web service composition at functional level a success.

In the following we will investigate on a novel FLC technique that fulfil such requirements. Not surprisingly, our approach will have roots in both AI planning systems and matchmaking based approaches to satisfy the latter requirements by:

i) considering an automated process to compute Web service composition. This requirement $R^{Composition}_{Automation}$ is studied by two different processes in Chapter 4.

ii) considering world-altering services , but also information-providing services to gather relevant information, and of course a mix of both. Therefore input, output, preconditions and effects will be considered to compute compositions of services. Moreover the number of functional parameters of the latter services will not be limited, or only in cases of performance research[5]. This requirement $R^{Service}_{Expressivity}$ is studied in Chapters 3 and 4.

iii) returning compositions as a partial order of relevant services which supports not only sequential ordering but also conditional choice and concurrency. This requirement $R^{Composition}_{Expressivity}$ is studied in Chapter 4.

iv) considering semantic connection between web services (through their input and output parameters) as a key concept to perform functional level composition. In a nutshell the technique relies on domain-dependent ontology for calculating semantic similarity scores (by means of standard and non standard matchmaking functions studied in Section 1.2.3) between the concepts in service descriptions, and applies this score to guide the searching process of the planning algorithm. This requirement $R^{Semantic}_{Composability}$ is studied in Chapter 3 and Section 4.1.

v) considering causal laws between Web services to achieve composition at functional level. This requirement $R^{Causal}_{Composability}$ is studied in Section 4.2.

vi) returning composition of services on the fly. Contrary to many approaches (e.g., [191, 205]) we will not assume any required decompositions, and build semantic connections (more or less methods and operators of SHOP2) of a composition on the fly. This requirement $R_{Flexibility}$ is studied in Chapter 3.

vii) presenting a general and extensible model to evaluate quality of both elementary and composition of semantic connections. From this, we introduce a global semantic connection selection based approach to compute the optimal composition. Therefore we provide a means

[5]It is obvious that the number of services together with their number of inputs, outputs, preconditions and effects have a direct impact on the performance of composition algorithms. However, to the best of our knowledge, no work focus on a trade-off between service expressivity and composition performance (Chapter 7).

to value "*Semantic*" web service composition. This requirement $R_{Optimization}$ is studied in Chapter 5.

viii) supporting standard description of Web services. This requirement $R^{Service}_{Applicability}$ is studied in Chapter 7.

ix) producing an abstract composition result as a reusable specification we can adapt for interfacing with standard proposals of web services orchestration such as OWL-S, BPEL4WS. This requirement $R^{Composition}_{Applicability}$ is studied in Chapter 7.

In the rest of the Ph.D report, we suggest to answer to Question Q_i presented in Introduction of this thesis report. The rest of the Ph.D report is structured as follows (see Figure 2.4).

Contribution

First of all, we focus on the main contributions of this thesis (i.e., Part II).

Chapter 3

Q1. How can the semantic composability of Web Services be ensured?

- In **Chapter 3**, semantic dependences between Web services are defined as the main composabilty ($R^{Semantic}_{Composability}$) criteria to achieve Web service composition. Towards this issue we focus on functional input and output parameters of services ($R^{Service}_{Expressivity}$).

Q2. What kind of properties the semantic composability of Web services have to satisfy?

- Contrary to most of approaches presented in Tables 2.2 and 2.1 we will consider more semantic matchmaking levels to direct semantic relations between services i.e., Subsumption, Intersection [124], Abduction [55], Concept Difference [198].

  From this innovative valuation of semantic connections between services, new issues related to robustness connections have been identified. Therefore our proposal will be also characterized by the fact that semantic robustness of any composition is checked during the composition process, which is unique to our proposal.

Q3. How to define a formal and flexible model that ensures correct semantic composability of Web services and easily support basic control flow of composition?

- In Chapter 3 a formal and flexible model ($R_{Flexibility}$) for Web service composition is studied. This model is introduced for facilitating computation (at Design Time) of semantic dependences between services and also their composition. This model aims at supporting expressive compositions of Web services ($R^{Composition}_{Expressivity}$).

Chapter 4

Chapter 4 deals with the problem functional level composition. Two complementary automated Web service composition approaches ($R^{Composition}_{Automation}$) are presented.

Q4. How can we compose Web services with semantic links?

- In Chapter 4 the first approach computes compositions depending on the following level of Web service description: input and output parameters of Web services (a part of $R^{Service}_{Expressivity}$). In this direction this approach considers semantic dependences as a composability criterion ($R^{Semantic}_{Composability}$). The flexibility ($R_{Flexibility}$) of the approach is ensured by the model presented in Chapter 3.

Q5. How can we compose semantic Web services wherein conditions on their functional parameters hold?

- In Chapter 4 the second approach computes compositions depending on the following level of Web service description: input, output parameters, preconditions and effects of Web services ($R^{Service}_{Expressivity}$). In this direction this second approach works with semantic dependences together with causal laws ($R^{Semantic}_{Composability}$ and $R^{Causal}_{Composability}$). The flexibility ($R_{Flexibility}$) of the approach is ensures by semantic links and causal laws of the problem.

Both approaches support expressive compositions of Web services ($R^{Composition}_{Expressivity}$), but with different levels. AI principles and matchmaking based systems are coupled to solve such a problem.

Chapter 5

Q6. How can we compare and select among a huge number of composition solutions that achieve the same goal?

- Contrary to approaches that compute optimal composition by considering only non functional parameters (e.g., quality of service), **Chapter** 5 focuses on functional parameters of Web services to value *semantics* of compositions. In particular this Chapter aims at computing semantic dependences based optimal Web service composition ($R_{Optimization}$). Starting from an initial set of web services, the goal of this chapter aims at selecting web services and maximizing the overall quality of their inter-connections by means of their semantic dependences according to a goal to achieve. Composition results of Chapter 4 can be inputs of the optimization process. Therefore $R_{Expressivity}$, $R_{Composability}$ and $R_{Flexibility}$ are ensured.

Our Approach in Use

Then Part III presents the prototype implementation of the theoretical Part II, running on scenarios in use at France Telecom R&D.

Chapter 6

Q9. Are the suggested approaches running on real scenarios and scaling large use cases?

- **Chapter** 6 presents three different scenarios wherein our approach of semantic Web service composition has been integrated. These three scenarios respectively refer to a Telecommunication, an E-Tourism and E-HealthCare scenario.

Chapter 7

Q7. How can we achieve an end to end composition of Web services at functional level?

- **Chapter** 7 presents a reference architecture to achieve an end-to-end composition of Web services.

Q8. Are the suggested approaches implemented?

- **Chapter** 7 presents the implementation of our service composition system. Besides satisfying requirements of the formal model, the system satisfies the following requirements: $R^{Service}_{Applicability}$ and $R^{Composition}_{Applicability}$.

  Chapter 7 presents the empirical evaluation of the system performance on the three latter scenarios in Use.

Conclusion

Finally we conclude with the summary, summarizes the thesis, discusses its contributions and provides a discussion of the method and future directions for this work.

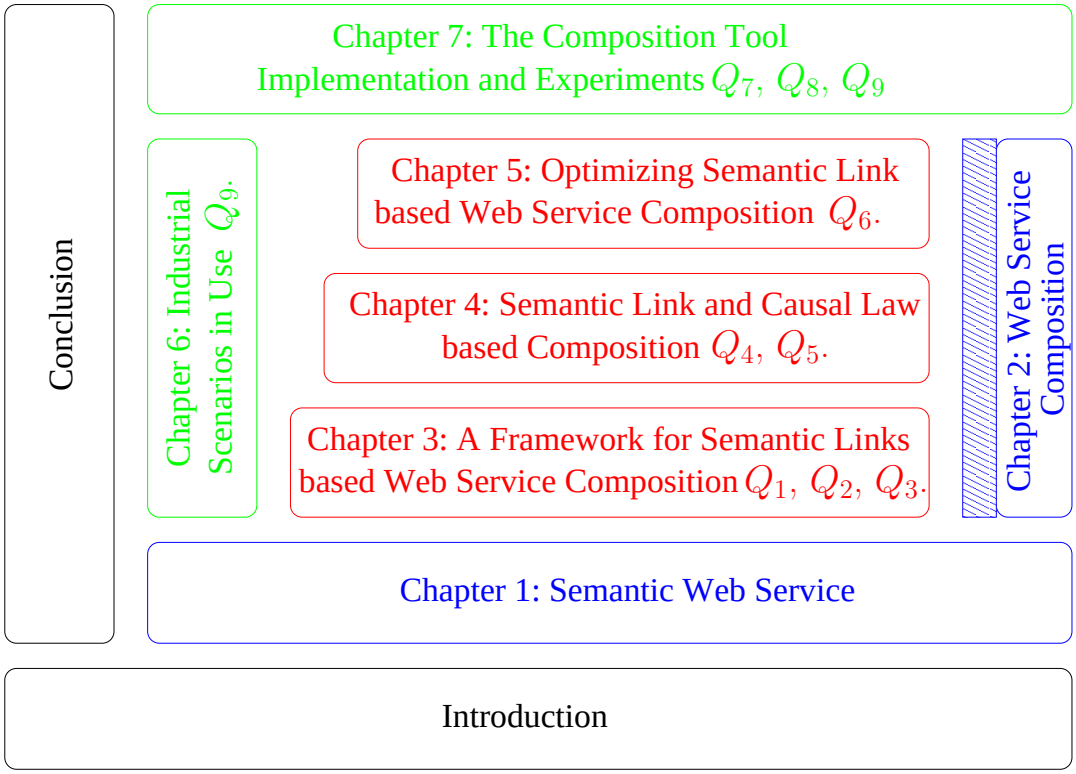


Figure 2.4: Organization and Inter Dependences of the Ph.D Report's Chapters (2).

Part II

Contribution

Chapter 3

A Framework for Semantic Links based Web Service Composition

In this Chapter, semantic dependences (redefined by **semantic links** in the Chapter) between Web services are defined as the main composabilty criteria to achieve Web service composition.

Towards this issue we focus on functional input and output parameters of services and assume that *any output parameter of any service can be semantically compared (e.g., trough subsumption, difference relationships, see Section 1.2.3) with any input parameter.*

To this end, functional input and output parameters of services are referred to concepts using a common ontology or Terminology $\mathcal{T}$[1] where the OWL-S service profile, WSMO service capability, SWSO Inputs/Outputs or SA-WSDL (see Section 1.3.2 for further details) can be used to describe them (through semantic annotations).

The **requirement** $R^{Service}_{Applicability}$ is address by the industrial applicability of our model to OWL-S, WSMO, SWSO or SA-WSDL. The **requirement** $R^{Semantic}_{Composability}$ is addressed by the **semantic links**. The **requirement** $R^{Service}_{Expressivity}$ is addressed by the **functional description of input and output parameters**.

Contrary to most of approaches presented in Tables 2.2 and 2.1 we will consider more semantic matchmaking levels to direct semantic relations between services i.e., Subsumption, Intersection [124], Abduction [55], Concept Difference [198].

From this innovative valuation of semantic links between services, new issues related to robustness links have been identified. Therefore our proposal will be also characterized by the fact that semantic robustness of any composition is checked during the composition process, which is unique to our proposal.

Moreover a formal and flexible model (**SLM i.e., Semantic Link Matrix**) for Web service composition is studied. This model is introduced for facilitating computation (at Design Time) of semantic links between services and also their composition. This model aims at supporting expressive compositions of Web services. On the one hand the **requirement** $R_{Flexibility}$ is addressed by the **SLM** model. On the other hand the **requirement** $R^{Composition}_{Expressivity}$ is addressed by

[1] Distributed ontologies is not considered here but is largely independent of the problem addressed in this Ph.D study. The interested readers may consult the following book [57] published recently to overcome issue related to distributed ontologies.

the expressive compositions (i.e., sequential, non determinist choice of Web services, concurrent) supported by the latter model.

The remainder of this Chapter is as follows. Section 3.1 describes in details semantic links between Web services. Section 3.2 presents the formal model SLM to deal with semantic Web service compositions though their semantic links. Section 3.3 describes how the latter model supports expressive Web service composition. Finally we conclude in Section 3.4.

3.1 Semantic Link

In this section we focus on semantic connections between Web services since they are considered as the main issue to form new value-added services by composition at functional level. These connections are required to semantically link output to input parameters (part of Requirement $R^{Service}_{Expressivity}$) of Web services (Requirement $R^{Semantic}_{Composability}$), and also to value Web service composition.

To this end we first introduce the definition of semantic links in Web service composition i.e., a formal concept for representing the latter semantic connections.

The remainder of this Chapter is as follows. First of all the definition of **semantic link** is introduced in Section 3.1.1. Then, we present how a semantic link can be semantically valued in Section 3.1.2. In Sections 3.1.3 and 3.1.4 we respectively highlight the issue related to validity and robustness of semantic links. Section 3.1.5 presents some approaches to ensure robustness of semantic links. Finally we synthesize in Section 3.1.6.

3.1.1 Definition

Here we address the composition problem as a discovery of semantic connections between Web services, which justify our focus on semantic links.

In the considered context retrieving a semantic connection between two Web services s_x and s_y is similar to discover a semantic similarity between an output parameter Out_s_y of s_y and an input parameter In_s_x of s_x (or vice versa). Consequently the goal is to find a matchmaking function between two knowledge representations encoded using the same ontology T.

Semantic links[2] (first introduced in one of our published work[3] [114]) between Web services will be in charge of valuating these semantic matchmaking functions. Therefore semantic links measure the quality of semantic links (i.e., quality of matchmaking functions) between Web services.

Definition 10. ***(Semantic Link)***
A semantic link $\langle s_y, Sim_{\mathcal{T}}(Out\_s_y, In\_s_x), s_x\rangle$ *is related to a logical dependency among an output* Out_s_y *and input parameter* In_s_x *of two different services* s_y *and* s_x*.*

Roughly speaking a semantic link (Figure 3.1) describes a semantic relation between an output parameter $Out\_s_y \in \mathcal{T}$ of a Web service s_y and an input parameter $In\_s_x \in \mathcal{T}$ of a Web service s_x. Thereby s_x and s_y are semantically and partially linked according to a matchmaking function $Sim_{\mathcal{T}}(Out\_s_y, In\_s_x)$. By definition a semantic link $\langle s_y, Sim_{\mathcal{T}}(Out\_s_y, In\_s_x), s_x\rangle$ implies that

[2] In the research area of AI planning, other kinds of links, well known as causal links or **protection intervals** [178, 132] are computed to perform planning.

[3] In this work the term *Semantic Link* refers to the *Causal Link* term, first introduced by [114] in the research domain of Web services. Their semantics are the same, only the terminology has changed from the previous work.

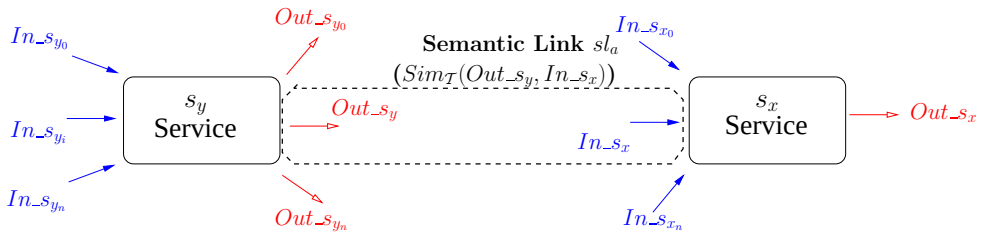


Figure 3.1: Illustration of a Semantic Link.

i) s_y precedes s_x since an output of s_y is exploited by an input of s_x; and ii) no Web service is interleaved between s_x and s_y. The matchmaking function $Sim_{\mathcal{T}}$ informs about the matchmaking type between the two parameters as concepts $Out\_s_y, In\_s_x \in \mathcal{T}$.

Moreover the latter function is useful not only to value the possible semantic connections between two Web services but also to compare them. Indeed it is obvious that some matchmaking types will be preferred and other will be disregarded with respect to their semantic quality. Let s_y and s_z be two Web services with their respective output parameters Out_s_y and Out_s_z. Suppose s_x such that Out_s_y and Out_s_z semantically match with In_s_x, $Sim_{\mathcal{T}}$ is required to value the two connections (Out_s_y, In_s_x) and (Out_s_z, In_s_x) and also to order them.

Example 8. ***(Semantic Link Illustration)***
Suppose `AdslEligibility` *(ELIG) and* `VoiceOverIP` *(VOIP) be two Web services introduced in Example 4 of Section 1.3.1. We assume without loss of generality that the* `AdslEligibility` *service precedes the* `VoiceOverIP` *service. According to the previous definition the two latter Web services are connected by two semantic links i.e.,*

(i) sl_a *described by* $\langle ELIG, Sim_{\mathcal{T}}(NetworkConnection, SlowNetworkConnection), VOIP\rangle$ *(Figure 3.2(a));*

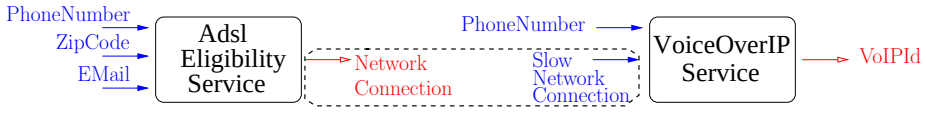
(ii) (sl_b *described by* $\langle ELIG, Sim_{\mathcal{T}}(NetworkConnection, PhoneNumber), VOIP\rangle$ *(Figure 3.2(b)).*

By (i) and (ii) two semantic connections valued by $Sim_{\mathcal{T}}$ *between an output parameter of* `AdslEligibility` *service and an input parameter of* `VoiceOverIP` *service are conceivable. However it is obvious that the semantic links* sl_a *and* sl_b *are not valued by the same matchmaking function with respect to the domain ontology* $\mathcal{T}$ *and TBox* $\mathcal{T}$ *(Figure 1.4).*

3.1.2 Semantic Link Valuation and Properties

Despite some existing methods [152, 124], solving a mapping problem is hard because the syntactic form of two knowledge representations rarely matches exactly. Indeed the semantic matchmaking $Sim_{\mathcal{T}}$ computed between an output and input parameter does not always refer to an Exact match e.g., sl_a in the previous example.

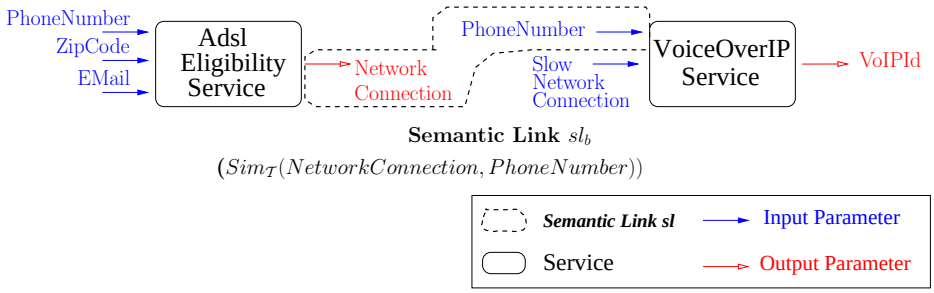
That is why $Sim_{\mathcal{T}}$ aims at expressing which matching type is used to chain Web services. As previously presented in Section 2.1, many Web service composition such as [210, 191, 45, 174, 40] reduced this function to the four well known matchmaking functions introduced by [152] (and presented in Section 2.1.2) with the extra match level Intersection of [124]:



Semantic Link sl_a

$(Sim_{\mathcal{T}}(NetworkConnection, SlowNetworkConnection))$

(a) A Semantic Link sl_a from the Motivating Example.



Semantic Link sl_b

$(Sim_{\mathcal{T}}(NetworkConnection, PhoneNumber))$

(b) A Semantic Link sl_b from the Motivating Example.

Figure 3.2: Illustration of a Semantic Link.

- **Exact** ($\equiv$) If the output parameter Out_s_y of s_y and the input parameter In_s_x of s_x are equivalent concepts; formally, $\langle \mathcal{T}, \mathcal{A} \rangle \models Out\_s_y \equiv In\_s_x$.
- **PlugIn** ($\sqsubseteq$) If Out_s_y is sub-concept of In_s_x; formally, $\langle \mathcal{T}, \mathcal{A} \rangle \models Out\_s_y \sqsubseteq In\_s_x$.
- **Subsume**($\sqsupseteq$) If Out_s_y is super-concept of In_s_x; formally, $\langle \mathcal{T}, \mathcal{A} \rangle \models Out\_s_y \sqsupseteq In\_s_x$.
- **Intersection** ($\sqcap$) If the intersection of Out_s_y and In_s_x is satisfiable; formally, $\langle \mathcal{T}, \mathcal{A} \rangle \not\models Out\_s_y \sqcap In\_s_x \sqsubseteq \bot$.
- **Disjoint** ($\bot$) Otherwise Out_s_y and In_s_x are incompatible i.e., $\langle \mathcal{T}, \mathcal{A} \rangle \models Out\_s_y \sqcap In\_s_x \sqsubseteq \bot$.

For instance, the PlugIn match means that an output parameter of a service s_y is subsumed by an input parameter of the succeeding service s_x whereas the Subsume match means that an output parameter of a service s_y subsumes an input parameter of the succeeding service s_x.

In the following the five previous semantic matching functions considered to value semantic links will be ordered to ease their comparison. Such an ordering is required to infer *which semantic matching function i.e., semantic link is more general or specific than another.* To do this we suggest to first order them with the logical implication operator $\Rightarrow$ as presented in theorem 1. Second the semantic matching functions will be discretized according to the previous partial order (Table 3.1).

The discretization of the matchmaking function enables us to estimate and represent the semantic quality of a semantic links.

Theorem 1. ***(Partial Order on Semantic Matchmaking Functions)***
The partial order on the matchmaking functions Exact, PlugIn, Subsume, Intersection is defined by the relations (i) and (ii) where $\Rightarrow$ refers to the binary and logical implication between $Out\_s_y \backslash_{set} \{\bot\}$ and $In\_s_x \backslash_{set} \{\bot\}$[4]

(i) $Exact \Rightarrow PlugIn \Rightarrow Intersection$

(ii) $Exact \Rightarrow Subsume \Rightarrow Intersection$

Proof. The proof of this theorem is divided into four different steps. Each step follows a trivial logical implication. □

Match Type	Logic meaning	Discrete $Sim_{\mathcal{T}}(Out\_s_y, In\_s_x)$
Exact ($\equiv$)	$\langle \mathcal{T}, \mathcal{A} \rangle \models Out\_s_y \equiv In\_s_x$	1
Plug-in ($\sqsubseteq$)	$\langle \mathcal{T}, \mathcal{A} \rangle \models Out\_s_y \sqsubseteq In\_s_x$	$\frac{3}{4}$
Subsume ($\sqsupseteq$)	$\langle \mathcal{T}, \mathcal{A} \rangle \models Out\_s_y \sqsupseteq In\_s_x$	$\frac{1}{2}$
Intersection ($\sqcap$)	$\langle \mathcal{T}, \mathcal{A} \rangle \not\models Out\_s_y \sqcap In\_s_x \sqsubseteq \bot$	$\frac{1}{4}$
Disjoint ($\bot$)	$\langle \mathcal{T}, \mathcal{A} \rangle \models Out\_s_y \sqcap In\_s_x \sqsubseteq \bot$	0

Table 3.1: Discretization of Semantic matching functions described by $Sim_{\mathcal{T}}$.

The function of matchmaking described by $Sim_{\mathcal{T}}(Out\_s_y, In\_s_x)$ between an output parameter Out_s_y of s_y and an input parameter In_s_x of s_x is close to the $degreOfMatch(Out\_s_y, In\_s_x)$ function introduced by [152].

However this function is considered in Web service composition and not in discovery. Moreover $Sim_{\mathcal{T}}$ is extended with the Intersection matchmaking function to consider more degrees of semantic matching.

The suggested approach introduced also a partial order based on the logical implication relation to compare semantic links and their values.

3.1.3 Semantic Link Validity

According to the previous matchmaking functions, a valid semantic link is defined by means of Definition 11.

Definition 11. ***(Valid Semantic Link)***
A semantic link $\langle s_y, Sim_{\mathcal{T}}(Out\_s_y, In\_s_x), s_x \rangle$ is valid iff $Sim_{\mathcal{T}}(Out\_s_y, In\_s_x) \neq Disjoint$ i.e., $\langle \mathcal{T}, \mathcal{A} \rangle \not\models Out\_s_y \sqcap In\_s_x \sqsubseteq \bot$.

Roughly speaking, a valid semantic link between two Web services describes a potential match [49] between two parameters (i.e., $\langle \mathcal{T}, \mathcal{A} \rangle \not\models Out\_s_y \sqcap In\_s_x \sqsubseteq \bot$), hence a valid and potential link between these Web services to form a composition.

Definition 12. ***(Valid Web Service Composition)***
A valid composition of Web services consists of only valid semantic links.

Example 9. ***(Valid Semantic Link Illustration)***
Consider the previous example with the semantic links sl_a and sl_b. It is trivial that the matchmaking function referred by sl_a is Subsume whereas the matchmaking function referred by sl_b is Disjoint. Indeed

[4] Since $\forall out\_s_y, \bot \sqsubseteq out\_s_y \not\Rightarrow \bot \sqcap out\_s_y$, we required to consider any out_s_y different from $\bot$ in our theorem. In the same way any In_s_x is different from $\bot$.

- $\langle \mathcal{T}, \mathcal{A} \rangle \models NetworkConnection \sqsupseteq SlowNetworkConnection;$
- $\langle \mathcal{T}, \mathcal{A} \rangle \models NetworkConnection \sqcap PhoneNumber \sqsubseteq \bot,$

with respect to the domain ontology $\mathcal{T}$.

According to definition 11 sl_a *is a valid semantic link whereas* sl_b *is not.*

3.1.4 Semantic Link Robustness

The five match levels are far from enough to bring Web service composition as a semantic links composition to its full potential.

The Exact match is clearly appropriate to chain two Web service parameters since they refer to equivalent concepts. The PlugIn match is also a possible match to plug an output parameter in an input parameter of another Web service since the output parameter provides more information than the input parameter required. The Disjoint match informs about the incompatibility of two Web service parameters hence an invalid semantic link.

Even if the matchmaking Exact, PlugIn, and Disjoint can be used without any change to value semantic links in a Web service composition, the match levels Intersection and Subsume need some refinements to be fully efficient for semantic links composition. Suppose a semantic link $\langle s_y, Sim_{\mathcal{T}}(Out\_s_y, In\_s_x), s_x \rangle$ valued by a Subsume match level i.e., $\langle \mathcal{T}, \mathcal{A} \rangle \models Out\_s_y \sqsupseteq In\_s_x$.

It is obvious that such a semantic link should not be directly applied in a Web service composition since the output parameter Out_s_y is not specific and specified enough to be exploited by the input parameter In_s_x. We say also that In_s_x is over specified and Out_s_y is under specified. In other words the output parameter Out_s_y requires an *Extra Description* to obtain a composition of these two Web services.

In the same way a semantic link valued by an Intersection match needs a comparable refinement. That is, two different kinds of semantic links require more attention: robust and non robust semantic links.

Definition 13. ***(Robust Semantic Link)***
A semantic link $\langle s_y, Sim_{\mathcal{T}}(Out\_s_y, In\_s_x), s_x \rangle$ *is robust if and only if* $Sim_{\mathcal{T}}(Out\_s_y, In\_s_x)$ *is either Exact or PlugIn.*

Example 10. ***(Some Limits of standard Matching functions)***
Let sl_a *illustrated in Figure 3.2(a) be the valid semantic link defined by:*

- $\langle ELIG, Sim_{\mathcal{T}}(NetworkConnection, SlowNetworkConnection), VOIP \rangle$

By definition 13 the valid semantic link sl_a *is not robust (Figure 3.3). Indeed the match level of* sl_a *is Subsume since*

- $\langle \mathcal{T}, \mathcal{A} \rangle \models NetworkConnection \sqsupseteq SlowNetworkConnection.$

It is obvious that this semantic link cannot be applied in a composition since the output parameter `NetworkConnection` *is under specified to be exploited by the input parameter* `SlowNetworkConnection`. *The output parameter* `NetworkConnection` *requires an* Extra Description *to enable a composition of these two services.*

Once the definition of robust semantic links introduced the notion of robust composition of Web services is as follows.

Definition 14. ***(Robust Web Service Composition)***
A composition of Web services is robust if and only if all its semantic links are robust.

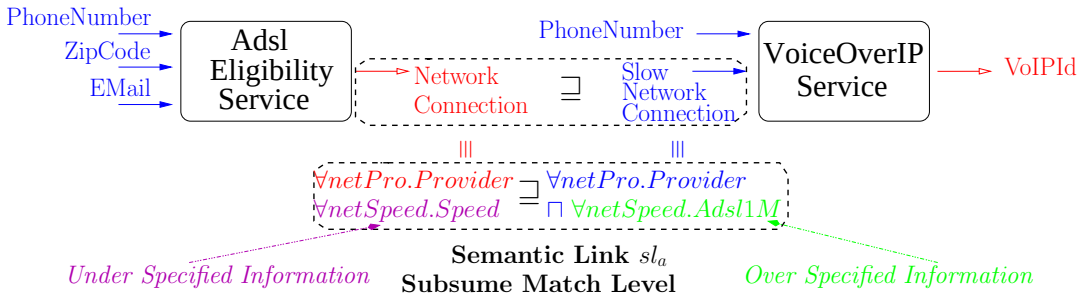


Figure 3.3: Illustration of a Non Robust Semantic Link sl_a valued by a Subsume Match Level.

Robust semantic links in a Web service composition are key components since they enable us to obtain robust Web service composition. In the opposite Web service composition with non robust semantic links can fail since some information is missing.

3.1.5 Ensuring Robustness in Semantic Links

A possible way to state the problem we obtain for non robust semantic links $\langle s_y, Sim_{\mathcal{T}}(Out\_s_y, In\_s_x), s_x\rangle$ valued with an Intersection or a Subsume match level is to find the information contained by the input parameter In_s_x and not by the output parameter Out_s_y. In other words some descriptions have to be retrieved to transform a non robust semantic link in its robust form.

To do this, we exploit a non-standard inference match level for DLs i.e., the difference (well known as subtraction operation) introduced by [35] for comparing DL descriptions and adapt it to the problem of semantic matching between Web service parameters.

The difference operator, introduced by [102] and explained in Section 1.2.3, enables us to remove from a given description all the information contained in another description. The difference between two concept descriptions C and D with $C \sqsubseteq D$ is given by (3.1).

$$C \backslash D := \min_{\preceq_d}\{B | B \sqcap D \equiv C \sqcap D\} \tag{3.1}$$

In other words, B represents an explanation on why D is not classified by C with respect to $\mathcal{T}$. Thus they defined the difference between two (in)comparable concept descriptions C and D as (3.1).

Remark 1. *Considering an $\mathcal{ALN}$ DL, Concept Abduction [48, 55] is also able to compute a concept expression B representing what is underspecified in D in order to completely satisfy C taking into account the information modelled in a TBox $\mathcal{T}$.*

As suggested and argued in Section 1.2.3 we focus on (3.1) to define Difference in $\mathcal{ALE}$ descriptions logics. The latter difference operator will be used to overcome the problem of robustness in Web service composition.

In this direction the Concept Difference is primarily performed to capture in a logical way the reason why an output parameter Out_s_y of a s_y and an input parameter In_s_x of s_x may not be chained by a robust semantic link.

The idea behind our approach is the following. In case a semantic link $\langle s_y, Sim_{\mathcal{T}}(Out\_s_y, In\_s_x), s_x\rangle$ is valid (not valued by a Disjoint matchmaking) but not robust (i.e., valued by neither an

Exact nor a PlugIn matchmaking), we compare Out_s_y and In_s_x to obtain two kinds of information:

(i) the *Extra Description* $In\_s_x \backslash Out\_s_y$ that refers to information required but not provided by Out_s_y in order to semantically link it to the input In_s_x of s_x;

(ii) the *Common Description* $Out\_s_y \sqcap In\_s_x$ that refers to information required by In_s_x and effectively provided by Out_s_y.

As previously said semantic links concerned by the computation of concept difference are links valued by either a Subsume or an Intersection matching i.e., non robust semantic links. In the Subsume case we compute the *Extra Description* B contained in In_s_x such that the matching between $B \sqcap Out\_s_y$ and In_s_x be Exact. By (3.1) this *Extra Description* $In\_s_x \backslash Out\_s_y$ is defined by $\min_{\preceq_d}\{B | B \sqcap Out\_s_y \equiv In\_s_x\}$ since $Out\_s_y \sqsupseteq In\_s_x$. In case the semantic link is valued by an Intersection match (i.e., $\neg(Out\_s_y \sqcap In\_s_x \sqsubseteq \bot)$) we compute the *Extra Description* B that is not specified in Out_s_y to reach a PlugIn match between $B \sqcap Out\_s_y$ and In_s_x.

Example 11. ***(Extra Description Illustration)***
Let sl_b illustrated in Figure 3.3 be the non robust semantic link defined by:

- $\langle ELIG, Sim_{\mathcal{T}}(NetworkConnection, SlowNetworkConnection), VOIP\rangle$.

Such a semantic link requires a semantic refinement to be robust enough in order to be applied in a composition of Web services. On the one hand the description missing in NetworkConnection to be plugged in the input parameter SlowNetworkConnection is referred by the Extra Description *i.e., $SlowNetworkConnection \backslash NetworkConnection$ i.e., $\forall netSpeed.Adsl1M$. On the other hand the common description defined by the conjunction of the output parameter of ELIG and the input parameter of VOIP is referred by the information required by $SlowNetworkConnection$ and effectively provided by $NetworkConnection$. In this way we remark that the intersection between the output parameter $NetworkConnection$ and the* Extra Description *$SlowNetworkConnection \backslash NetworkConnection$ i.e., $\forall netSpeed.Adsl1M$ is an Exact match with* `SlowNetworkConnection`.

Example 12. ***(Common Description Illustration)***
Let sl_b be the non robust semantic link illustrated in Figure 3.3. The Common Description *is not empty since this is defined by $SlowNetworkConnection \sqcap NetworkConnection$ i.e., $\forall netPro.Provider \sqcap \forall netSpeed.Speed$.*

We illustrated the rationale of our approach by computing what is required in order to replace a non robust semantic link by its robust form. In particular, we are able to change an Intersection by a PlugIn match, and a Subsume by an Exact match in order to obtain robust semantic links (modelled with ✓in Table 3.2). We could also consider other substitutions of matchmaking functions e.g., find a way to change a Subsume by a PlugIn match, or an Intersection by an Exact match and so on. However these substitutions are out of interest in Web service composition since

(i) some substitutions required the computation of $Out\_s_y \backslash In\_s_x$ (modelled with $\boldsymbol{\times}_{i)}$ in Table 3.2) e.g., from PlugIn to Exact;

(ii) some others are not relevant because they implied a loss of matchmaking quality (modelled with $\boldsymbol{\times}_{ii)}$ in Table 3.2) e.g., from PlugIn to Subsume.

Table 3.2 summarizes these different levels of substitution. Suppose the substitution of a PlugIn by an Exact match in order to improve a semantic link valued by a PlugIn match level.

The case under consideration is i) since we have to compute B' such that $Out\_s_y \equiv B' \sqcap In\_s_x$. B' is defined by $Out\_s_y \backslash In\_s_x$ to model the exact match. Unfortunately the description B' cannot be added to In_s_x since input parameters of services are supposed static without possible alteration. In the opposite output parameters of services may be enhanced by some *Extra Descriptions*[5] ($Out\_s_y \sqcap B$) in order to be chained with input parameters of other services. Now suppose the case ii) wherein the Subsume match is replaced by a PlugIn match. Consequently, the *Extra Description* B' is computed as $B' \sqsubset B$ such that B is defined by $In\_s_x \backslash Out\_s_y$. By the way $B' \sqcap Out\_s_y \sqsubseteq In\_s_x$ whereas $B \sqcap Out\_s_y \equiv In\_s_x$. It is obvious that B is more appropriate than B'. The former enables an Exact match whereas the latter changes the Subsume by the PlugIn match.

Substituted Match Type	**Potential Substitute Match Type**			
	Exact	PlugIn	Subsume	Intersection
Exact	-	✗$_{ii)}$	✗$_{ii)}$	✗$_{i),ii)}$
PlugIn	✗$_{i)}$	-	✗$_{i),ii)}$	✗$_{i)}$
Subsume	✓	✗$_{ii)}$	-	✗$_{ii)}$
Intersection	✗$_{i)}$	✓	✗$_{i)}$	-

Table 3.2: Substitution of Match Levels for Web Service Composition. Legend: ✗ = not supported, ✓= fully supported.

In case some semantic links $\langle s_y, Sim_{\mathcal{T}}(Out\_s_y, In\_s_x), s_x \rangle$ are not robust enough but valid, we are able to compute an *Extra Description* from In_s_x in order to substitute the previous link by its robust form. In other words semantic links valued by a Subsume or an Intersection match level move to robust semantic links in case their *Extra Descriptions* are provided. The latter description is essential to compute a robust service composition. The *Extra Description* returned by difference (3.1) is not only necessary to explain where a semantic link composition may fail but also why a semantic link failed and how to improve it. A composition failure is due to non robust semantic links since the matchmaking between Web services parameters is not robust enough.

3.1.6 Synthesis

Limitation of Semantic Links

In computing semantic link, a single output from one service can be related to only a single input to a possible following service (see Definition 10 of semantic link). In our approach this is a necessary restriction. It appears that our use of DL for describing inputs and outputs would allow generalizations of inputs and outputs to be matched. For instance a conjunction of output parameters from different Web services could be semantically matched to one (or more) input parameter(s) of a (or more) services. To this end the semantic link definition requires to be extended. Such an extension is straightforward.

[5] As we will study more precisely in Section 4.1, the *Extra Description* could be known, for instance, i) by discovering Web services that provide such a description as output parameter or ii) by requesting the end-user to provide such a description.

Some Concluding Remarks

Till now, we have specified the formal context for semantic Web service composition at functional level. In this direction semantic links (Requirement $R^{Semantic}_{Composability}$) between functional input and output parameters of Web services (Requirement $R^{Service}_{Expressivity}$) have been introduced, defined and illustrated on several semantic Web services. The latter links are considered as a key concept to form compositions of semantic Web services.

The main difference with techniques described in Section 2.1 is related to the semantic level of potential links between Web services. Contrary to [139] that considers preconditions and effects of Web services, here we focus more on utilization of output parameters by input parameters of other Web services by means of semantic links. Moreover the semantic links consider different semantic levels (Equivalence, Subsumption, Intersection[124], Abduction [55], Difference [198]) of direct relations between Web services whereas [139, 140] check only Satisfiability between some effects and preconditions of Web services.

Moreover we presented important criteria related to validity and robustness of semantic links. Such criteria can be used to detect mismatching between Web services. More precisely valid semantic links refer to semantic links that can be useful for any Web service composition whereas robust semantic links are required to form robust Web service composition. In addition we addressed different solutions to ensure robustness of semantic links such as the Concept Abduction or Concept Difference. Therefore mismatched connections between Web services can be repaired through a fine-grained process of mediation (actually finer that the WSMO mediation).

From these definitions we will define a composition of Web services at functional level as

> *A plan of services wherein all services are semantically well ordered and well linked by semantic links (robust or not but valid).*

3.2 Semantic Link Matrix

In this section, we introduce a formal model so called *Semantic Link Matrix* (henceforth SLM). Such a model aims at easing the automated process of Web service composition and at improving its performance i) by computing, ii) by storing and iii) by classifying all *valid semantic link* (Requirement $R^{Semantic}_{Composability}$) between a set of Web services in a simple, intuitive and flexible way (Requirement $R_{Flexibility}$).

In this section we do not address Web service composition but the problem of:

> *Elaborating and computing a formal and flexible model which consists of relevant, precomputed and valid semantic links we can potentially find in a composition.*

The rest of this Section is organized as follows. Section 3.2.1 introduces important notations related to the formal model SLM and gives a detailed definition of the model. The construction of the SLM is detailed in Section 3.2.2. Finally we synthesizes in Section 3.2.3.

3.2.1 Notations and Definition of Semantic Link Matrix

The Semantic Link Matrix, introduced in this section, aims at suggesting a model for composing a finite set of semantic Web services S_{Ws}.

To this end the latter set of services is supposed to be discovered in a relevant way, given a composition goal. In other words we assume that a process of discovery such as [197, 24] has been performed in order to first retrieve a finite set of Web services S_{Ws}.

Given this set of Web services, we aim at first computing a formal model to organize these Web services (Section 3.2) and then using this model to achieve a composition of a subset of the latter discovered services (Chapter 4).

In the following we first give some notations and then we focus on the *Semantic Link Matrix*.

Some Notations

Since the definition of SLMs requires some new but simple definitions i.e., a) $Out(s_y)$, b) $In(s_y)$, c) $Input(S_{Ws})$, d) $Output(S_{Ws})$ and e) β, we suggest to introduce and illustrate them in the following.

Suppose S_{Ws} be the set of Web services with the upcoming services s_x and s_y. From this,

a) $Out(s_y)$ refers to the set of output parameters of the Web services s_y;

b) $In(s_y)$ is the set of input parameters of the Web services s_y;

c) $Input(S_{Ws})$ refers to the set of all input parameters of all services included in the set S_{Ws};

d) $Output(S_{Ws})$ refers to the set of all output parameters of all services included in the set S_{Ws}.

e) β will refer to the composition goal in the rest of the Ph.D report. In our approach β is simply viewed as a subset of the TBox $\mathcal{T}$. These concepts have to be reached i.e., our ultimate issue is to find a composition of Web services which are able to find an instance of each concept in β. β will be required and used during the AI planning-based composition (Section 4.1).

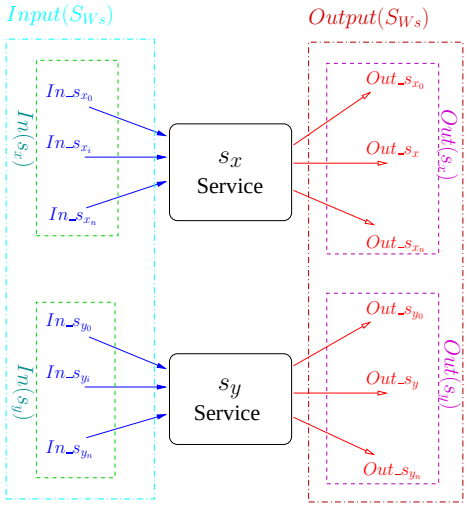


Figure 3.4: Some Important Notations required to formally define SLMs.

Example 13. *($Out(s_y)$, $In(s_y)$, S_{Ws}, $Input(S_{Ws})$ **and** $Output(S_{Ws})$)*
Let the motivating example 4 be in Section 1.3.1 and its set of relevant Web services S_{Ws} be defined by `AdslEligibility`, `VoiceOverIP`, `TvOverIP` *and* `LiveBox` *services. Table 3.3 and Figure 3.4 illustrates the previous defined elements related to S_{Ws}.*

Services s_x	`AdslEligibility` S_a			`VoiceOverIP` S_b	`TvOverIP` S_c	`LiveBox` S_d
	−		+			
$Input(s_x)$	$\{PhoneNum, ZipCode, Email\}$			$\{PhoneNum, SlowNetwork-Connection\}$	$\{PhoneNum, FastNetwork-Connection\}$	$\{PhoneNum, IPAddress, Decoder\}$
$Output(s_x)$	$\{Slow NC\}$	$\{NC\}$	$\{Fast-NC\}$	$\{VoIPId\}$	$\{VideoDecoder\}$	$\{Invoice\}$
$Input(S_{Ws})$	$\{PhoneNum, ZipCode, Email, SlowNetworkConnection, FastNetworkConnection, IPAddress, Decoder\}$					
$Output(S_{Ws})$	$\{Invoice, FastNetworkConnection, NetworkConnection, VoIPId, VideoDecoder, SlowNetworkConnection\}$					

Table 3.3: $Out(s_x)$, $In(s_x)$, $Input(S_{Ws})$ and $Output(S_{Ws})$ in the Motivating Example.

Definition of Semantic Link Matrix

In the following we define an SLM as a matrix containing all enabled, legal and valid transitions for a Web service composition goal. Non valid semantic links are disregarded from an SLM point of view since a semantic link valued by a Disjoint match level (Table 3.1) refers to an inconsistency between two output and input parameters of Web services. In contrary all valid semantic links between two output and input parameters of Web services are explicitly represented with a value pre-computed by means of the $Sim_{\mathcal{T}}$ function (Table 3.1). The latter value is based on the semantic quality of the valid semantic link. The more valid semantic links, the better the functional composition.

Definition 15. ***(Semantic Link Matrix SLM)***
The set of $p \times q$ Semantic Link Matrices[6] *is defined as matrices $\mathrm{M}_{p,q}(\mathcal{P}(S_{Ws} \times (0,1]))$. Their columns $c_{j,j\in\{1,\ldots,q\}}$ are labelled by concepts in $(Input(S_{Ws}) \cup \beta) \subseteq \mathcal{T}$ i.e., the inputs parameters of services $Input(S_{Ws})$ in S_{Ws} and the concepts described by the goal set $\beta \subseteq \mathcal{T}$. Rows $r_{i,i\in\{1,\ldots,p\}}$ are labelled by $Input(S_{Ws})$, the inputs parameters of services in S_{Ws}. Each entry $m_{i,j}$ of an SLM $\mathcal{M}$ is defined as a set of pairs $(s_y, score) \in S_{Ws} \times (0,1]$ such that*

$$(s_y, score) := (s_y, Sim_{\mathcal{T}}(Out\_s_y, c_j)) \;\; if \;\; s_y \in S_{Ws}, \; Out\_s_y \in Out(s_y) \tag{3.2}$$

with $r_i \in \mathcal{T} \cap In(s_y) \subseteq Input(S_{Ws})$ is the label of the i^{th} row.
with $c_j \in \mathcal{T} \cap (Input(S_{Ws}) \cup \beta)$ is the label of the j^{th} column.

A SLM is seen as a matrix with entries in $\mathcal{P}(S_{Ws} \times (0,1])$. Each entry of an SLM refers to a set of pairs $(s_y, score)$ such that the score refers to a semantic similarity $Sim_{\mathcal{T}}(Out\_s_y, c_j)$ between an output parameter $Out\_s_y \in \mathcal{T}$ of a service s_y and an input parameter $c_j \in \mathrm{Input}(S_{Ws}) \cup \beta$ of another service in S_{Ws}. Indeed columns c_j of the SLM are labelled by all input parameters of the relevant Web services S_{Ws}.

[6] $\mathcal{P}(S)$ refers to power set of S.

Remark 2. ***(SLM or a Matrix of Valid Semantic Links)***
Since all entries of SLMs are defined on $\mathcal{P}(S_{Ws} \times (0,1])$*, possible values of semantic links are defined in* $(0,1]$*. According to Table 3.1, the only possible matchmaking functions met by the semantic links are Intersection, Subsume, PlugIn and Exact. According to Definition 11, only valid semantic links can be referred in an SLM.*

Remark 3. ***(Key Feature of SLMs)***
The innovative feature of SLMs is to label rows and columns together with the same set of concepts in $\mathcal{T}$ *i.e., input parameters of Web services in* S_{Ws}*. The link between a row and column of such a matrix is defined by a possible semantic link between an output parameter of a service and a column of the matrix.*

All SLMs in $\mathrm{M}_{p,q}(\mathcal{P}(S_{Ws} \times (0,1]))$ of a given domain are defined by means of their number p of rows and number q of columns. SLMs of a domain are also related to the pre-defined goal β. By considering $\#(\beta)$ be the cardinality of goals we have the following relations:

$$p = \#(Input(S_{Ws})) \tag{3.3}$$
$$q = p + \#(\beta) - \#(\beta \cap Input(S_{Ws})) \tag{3.4}$$

Definition 16. ***(Dimension of SLMs)***
In compliance with [20] the dimension of a semantic link matrix in $\mathrm{M}_{p,q}(\mathcal{P}(S_{Ws} \times (0,1])$ *is defined by:*

$$dim_{\mathcal{P}(S_{Ws} \times (0,1])}\mathrm{M}_{p,q}(\mathcal{P}(S_{Ws} \times (0,1])) = p \times q. \tag{3.5}$$

In the general case, SLMs are not square matrices since $q > p$*.*

i/j index	1	2	3	4	5	6	7	8
$r_{i.label}$	Email	Decoder	FastNC	IPAddress	PhoneNum	SlowNC	ZipCode	
$c_{j.label}$	Email	Decoder	FastNC	IPAddress	PhoneNum	SlowNC	ZipCode	Invoice

Table 3.4: Labels of the rows r_i and columns c_j of the 7×8 matrix $\mathcal{M}$.

Example 14. ***(Illustration of SLMs indexes and labels)***
Suppose the motivating example 4 be in Section 1.3.1 with the following six Web services:

- `AdslEligibility` *(*S_a*),*
- `AdslEligibility-` *(*S_a^-*),*
- `AdslEligibility+` *(*S_a^+*),*
- `VoiceOverIP` *(*S_b*),*
- `TvOverIP` *(*S_c*),*
- `LiveBox` *(*S_d*),*

as elements of S_{Ws}*.*

Suppose $\{Invoice\}$ *be the only concept in goal* β*.*

The number of rows and columns are respectively equal to 7 and 8 (Table 3.4) according to equalities (3.3), (3.4) and definition of SLMs.

Therefore rows and columns of the SLM $\mathcal{M}$ *of this domain are respectively indexed by:*

- $\{1, ..., 7\}$, $\{1, ..., 8\}$,

and labelled by:

- *concepts* $r_{i,i\in\{1,...,7\}}$, $c_{j,j\in\{1,...,8\}}$ *of the TBox* $\mathcal{T}$.

For instance the row r_1 *and the column* c_1 *are labelled by the concept Email whereas the row* r_3 *and column* c_3 *are labelled by the concept FastNetworkConnection (Table 3.4).*

3.2.2 Construction of Semantic Link Matrices

The Algorithm 1 presents the different steps of the SLM construction [117]. Such a construction consists in discovering a semantic similarity *score* between the output parameters of all services $s_y \in S_{Ws}$ and the input parameters of another service in S_{Ws}. In case the value *score* is not null, the pair $(s_y, score)$ is added in the *SLM*. To do this, all input parameters of S_{Ws} are parsed two times (p times in line 4 and q times in line 6) to build the $p \times q$ matrix. Moreover all output parameters of S_{Ws} are parsed (line 8) to value the matchmaking function (Table 3.1) of a potential valid semantic link.

Algorithm 1: Semantic Link Matrix Construction.

```
1  Input: S_Ws, T.
2  Result: The Semantic link matrix M of the domain.
3  begin
4      foreach row r_i of the SLM M do
5          foreach column c_j of the SLM M do
6              foreach Web service s_y ∈ S_Ws do
7                  if r_i ∈ In(s_y) then
8                      if ∃Out_{s_y} ∈ Out(s_y) & Sim_T(Out_s_y, c_j) ≠ 0 then
9                          m_{r_i,c_j} ← m_{r_i,c_j} ∪ (s_y, Sim_T(Out_s_y, c_j));
10     return M;
11 end
```

According to Algorithm 1 the Semantic link matrix construction is mainly function of the cardinality of $Output(S_{Ws})$ and $Input(S_{Ws})$. The algorithmic complexity of the SLM construction is then

$$\theta(\#(Input(S_{Ws})) \times \#(Input(S_{Ws})) \times \#(S_{Ws})) \tag{3.6}$$

so cubic in the worst case [117]. However an optimal process of the SLM construction can be computed in

$$\theta(\#(Input(S_{Ws})) \times \#(Output(S_{Ws}))) \tag{3.7}$$

$$or\ \theta(Max\{\#(Input(S_{Ws})), \#(Output(S_{Ws}))\}^2) \tag{3.8}$$

so square in case $\#S_{Ws} \ll \#(Input(S_{Ws}))$.

Example 15. ***(Semantic link matrix illustration with Tables 3.3, 3.4)***
Suppose the motivating example with the set of Web services S_{Ws} *in Table 3.3. We can easily*

compute the SLM of the domain according to algorithm 1 and Table 3.4 to obtain an SLM $\mathcal{M}$ with entries in $\mathcal{P}(S_{Ws} \times \{\frac{1}{4}, \frac{1}{2}, \frac{3}{4}, 1\})$.

$$\mathcal{M} = \begin{pmatrix} \emptyset & \emptyset & \{(S_a^-,\frac{1}{2}),(S_a,\frac{1}{2}),(S_a^+,1)\} & \emptyset & \emptyset & \{(S_a^-,1),(S_a,\frac{1}{2}),(S_a^+,\frac{3}{4})\} & \emptyset & \emptyset \\ \emptyset & \emptyset & \emptyset & \emptyset & \emptyset & \emptyset & \emptyset & \{(S_d,1)\} \\ \emptyset & \{(S_c,\frac{3}{4})\} & \emptyset & \emptyset & \emptyset & \emptyset & \emptyset & \emptyset \\ \emptyset & \emptyset & \emptyset & \emptyset & \emptyset & \emptyset & \emptyset & \{(S_d,1)\} \\ \emptyset & \{(S_c,\frac{3}{4})\} & \{(S_a^-,\frac{1}{2}),(S_a,\frac{1}{2}),(S_a^+,1)\} & \{(S_b,\frac{1}{4})\} & \emptyset & \{(S_a^-,1),(S_a,\frac{1}{2}),(S_a^+,\frac{3}{4})\} & \emptyset & \{(S_d,1)\} \\ \emptyset & \emptyset & \emptyset & \{(S_b,\frac{1}{4})\} & \emptyset & \emptyset & \emptyset & \emptyset \\ \emptyset & \emptyset & \{(S_a^-,\frac{1}{2}),(S_a,\frac{1}{2}),(S_a^+,1)\} & \emptyset & \emptyset & \{(S_a^-,1),(S_a,\frac{1}{2}),(S_a^+,\frac{3}{4})\} & \emptyset & \emptyset \end{pmatrix}$$

The entry $m_{5,3}$ (i.e., $m_{PhoneNum,FastNC}$) is equal to $\{(S_a^-,\frac{1}{2}),(S_a,\frac{1}{2}),(S_a^+,1)\}$. Indeed a Web service S_a with one input parameter $PhoneNum$ (row 5) and an output $NetworkConnection$ semantically similar to $FastNetworkConnection$ (column 3) exists in S_{Ws}. $\langle S_a, Sim_{\mathcal{T}}(NetworkConnection, FastNetworkConnection), S_b\rangle$ is a valid semantic link since the matchmaking function valued by $Sim_{\mathcal{T}}(NetworkConnection, FastNetworkConnection)$ is a Subsume match level i.e., $\frac{1}{2}$.

Proposition 1. *An entry $m_{i,j}$ of a semantic link matrix $\mathcal{M} \in \mathrm{M}_{p,q}(\mathcal{P}(S_{Ws} \times (0,1]))$ is different from the empty set if and only if the following conditions is satisfied:*

- *$\exists s_y \in S_{Ws}$ with at least one input $r_{i.label} \in \mathcal{T}$ and one output $Out\_s_y \in Out(s_y) \cap \mathcal{T}$ such that $Sim_{\mathcal{T}}(Out\_s_y, c_{j.label}) \neq 0$ (SLM definition).*

3.2.3 Synthesis

Semantic Link Matrix and Scalability

Since the flexibility feature of Web service composition model is supposed to be fundamental in volatile environments such as the Web services area (Requirement $R_{Flexibility}$), the SLM model aims at being as flexible as possible.

Indeed dynamic process of Web service discovery could be applied to modify an SLM of given domain. In this direction the SLM model is quite appropriate to support fundamental criteria such as alteration and modification (e.g., insertion, deletion, or update) of Web services in S_{Ws}, hence a flexibility of the model. Each new update of S_{Ws} is supported by an SLM revision together with a revision of the domain ontology T. For instance the integration of a new Web service is related to the insertion of new labelled rows and columns in the worst case. In the alternative case the integration of a Web service means a simple insertion of this service in the relevant entry(ies) of the specific SLM.

Therefore incremental systems wherein new Web services are progressively added, are supported by the SLM model. In the same way deletion of Web services in a given SLM can be easily performed as well.

The set of Web services S_{Ws} involved in the SLM is closed in order to limit its dimension. This assumption seems appropriate to perform Web service composition with a bounded number of Web services such as required in real and industrial scenarios. Even if the SLM dimension and its computation is only square with the number of input parameters of S_{Ws}, such a level of complexity may be a real issue to scale with more complex cases of composition e.g., hundreds of thousands of Web services involved in a composition. Computing the SLM in such a domain can really weaken the model used by our composition approach, presented in Section 4.1. One possible direction to overcome this issue is to consider SLMs with a major part of empty entries since SLMs are sparse in most of cases. In such a case the SLM model is replaced by a more subtle model i.e.,

lists of semantic links wherein we store only valid semantic links. The space complexity of this model is then drastically reduced. However our composition approach presented in Section 4.1 will require some minor changes since our composition method assumes an SLM of the domain. We remark that scenarios that require a composition of hundreds of thousands of Web services is more than improbable in real industrial scenario since its execution will be a real open issue.

Goal-Independent Semantic Link Matrix

The Definition 15 can be detached to the concept of goal in order to provide a goal-independent SLM definition. In this direction we provide a context adapted for Web service composition and for no predefined goals. This is quite appropriate for adaptability of our model in any composition problem.

Some Concluding Remarks

In this section, we introduced a formal model so called *Semantic Link Matrix* and its construction process. Such a model aims at preparing the automated process of Web service composition i) by computing, ii) by storing and iii) by classifying all *valid semantic link* (Requirement $R^{Semantic}_{Composability}$) between a set of Web services in a simple, intuitive and flexible way (Requierement $R_{Flexibility}$).

The key contribution of the SLM is a formal and semantic model to control a set of Web services S_{Ws} which is relevant for a composition goal.

In the following section, we describe how the SLM of a given domain can support expressive Web service composition.

3.3 Semantic Link Matrix and Web Service Composition

The concept of *semantic link* introduced in Section 3.1 is quite appropriate to model trivial composition such as the composition illustrated in Figure 3.5[7].

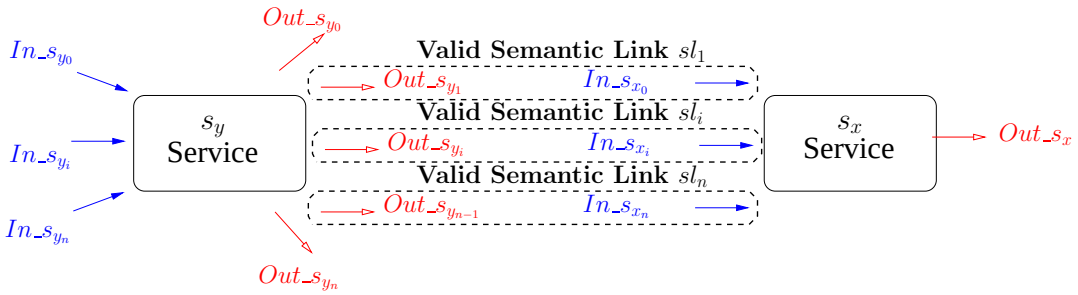


Figure 3.5: Illustration of the simplest case of a Web service composition $s_x \hat{\circ} s_y$.

[7]In such a level of composition, all input parameters of a Web service s_x can be provided by some output parameters of the direct Web service predecessor s_y.

Unfortunately Web service composition is more subtle than the latter trivial case of composition. Indeed, in real scenarios a composition of Web services can be designed by means of the simple sequence construct (Figure 3.5), but also by means of more complex constructs such as the non deterministic choice of Web service or the concurrency constructs (Requirement $R^{Composition}_{Expressivity}$) as well (Figure 3.6).

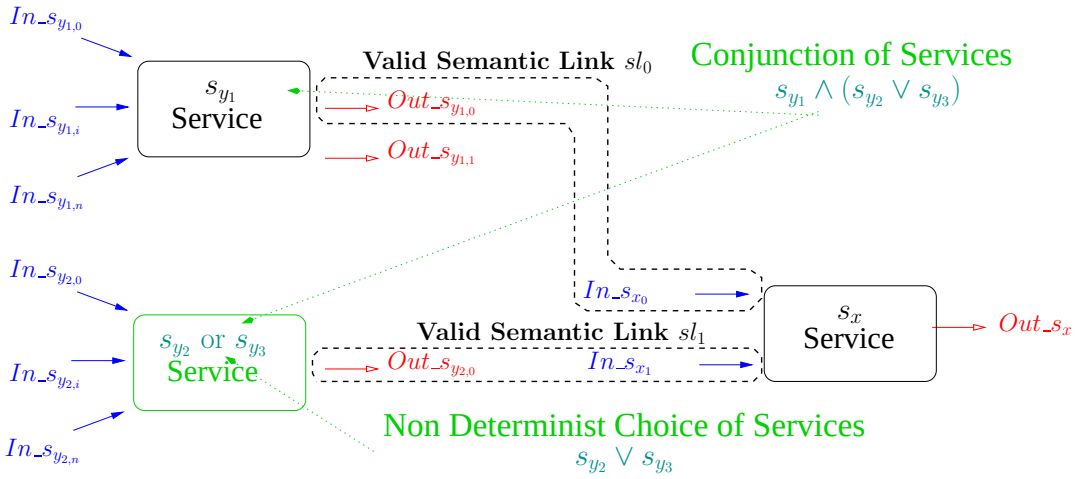


Figure 3.6: Illustration of more complex cases of a Web service composition.

By classifying all valid semantic links in the SLM (see Section 3.2), we are ensured to discover all compositions of Web services which are related by semantic links.

By introducing the SLM model we consider a simpler composition problem i.e., the semantic link composition. Therefore the Web service composition is mapped to a semantic link composition wherein semantic links inform about semantic connections between Web service. The solutions of Web service composition will be mainly oriented by the SLM of the domain.

In this section we describe how the latter model can support not only trivial composition (Figure 3.5) but also expressive compositions of Web services (Requirement $R^{Composition}_{Expressivity}$).

The rest of this Section is organized as follows. Section 3.3.1 introduced the concept of *Sequence Composability* between Web services. In Section 3.3.2 we describe how expressive Web service composition can be modelled by the semantic link matrix of a given domain. Finally Section 3.3.3 synthesizes the Section.

3.3.1 Sequence Composability of Web Services

It is straightforward to identify a semantic link as a very basic composition of two Web services since the latter link describes a simple sequence of two Web services. From this we can easily extend the basic composition to the trivial composition $s_x \hat{\circ} s_y$ wherein $s_x \hat{\circ} s_y$ (Figure 3.5) simply means that s_y precedes s_x and there exists a strictly positive value of $Sim_{\mathcal{T}}$ between each input parameter of s_x and some output parameters of s_y.

Such simple compositions come from a special case of the sequence composability definition introduced by [114].

Definition 17. ***(Sequence Composability)***
The sequence composability (Figure 3.7) between two Web services s_y and s_x is defined as a composition $s_x \circ s_y$ if and only if an output of s_y is exploited by an input of another Web service s_x.

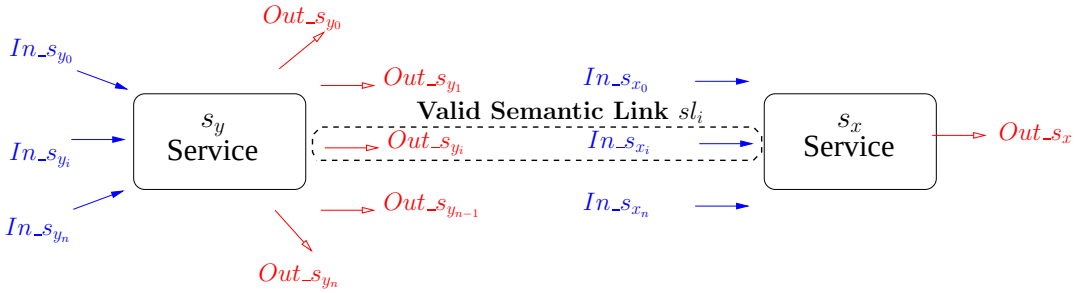


Figure 3.7: Illustration of a Sequence Composability of Services $s_x \circ s_y$.

3.3.2 Modelling Sequence Composability and Expressive Compositions in SLM

As previously mentioned basic and trivial compositions of Web services are the first and the most intuitive categories of compositions. That is why it seems relevant to make sure that SLMs are able to simply retrieve these models of compositions. In other words a SLM of a given domain have to satisfy the sequence composability definition introduced in the beginning of this section.

Theorem 2. ***(Sequence Composability and SLMs)***
Let $\mathcal{M}$ be an SLM, and s_x, s_y be two Web services in S_{Ws}. s_x and s_y are sequence-composable iff the following condition holds:

- $\exists i \in \{1,..,p\}$, $\exists j \in \{1,..,q\}$, $\exists v \in (0,1]$ *such that* $(s_y, v) \subseteq m_{i,j}$. $c_{j.label}$ *and* $r_{i.label}$ *are respectively inputs of* s_x *i.e.,* $In(s_x)$ *and* s_y *i.e.,* $In(s_y)$.

Proof. Consider the proof of theorem 1 as the following two implications.

($\Rightarrow$) Let s_x, s_y be two Web services in S_{Ws} and $\mathcal{M}$ be an SLM with entries in $\mathcal{P}((S_{Ws} \cup \mathcal{T}) \times (0,1])$. Moreover, we consider the *Sequence-composability* of s_x and s_y such that an output of the Web service s_y is consumed by the input of another Web service s_x i.e., $s_x \circ s_y$. According to the SLM definition, input parameters of s_x are labelled in $\mathcal{M}$ as concepts in $\mathcal{T}$. Thus we may suppose $\{1, ..., p_{s_x}\}$ as the index of the s_x input parameters in $\mathcal{M}$ without loss of generalities. According to the *Sequence-composability* definition, $\exists j \in \{1, ..., q_{s_x}\}$ such that $Sim_{\mathcal{T}}(Out\_s_y, c_{j.label}) > 0$ since an output $Out\_s_y \in Out(s_y)$ of one Web service s_y is consumed by an input $c_{j.label}$ of another web service s_x. Consequently $\langle s_y, Sim_{\mathcal{T}}(Out\_s_y, c_{j.label}), s_x \rangle$ is a valid semantic link. According to the property 1.i), an entry $m_{i,j}$ from $\mathcal{M}$ is different from the empty set. Finally $\exists i \in \{1, ..., p_{s_x}\} \subseteq \{1, ..., p\}, \exists j \in \{1, ..., q_{s_x}\} \subseteq \{1, ..., q\}$ such that $(s_y, Sim_{\mathcal{T}}(Out\text{-}\_s_y, c_{j.label})) \subseteq m_{i,j}$ with $c_{j.label} \in In(s_x)$ and $r_{i.label} \in In(s_y)$.

($\Leftarrow$) Suppose $\exists i \in \{1,...,p\}, \exists j \in \{1,...,q\}, \exists score \in (0,1]$ such that $(s_y, score) \subseteq m_{i,j}$ with $c_{j.label} \in In(s_x) \subseteq \mathcal{T}$ and $r_{i.label} \in In(s_y) \subseteq \mathcal{T}$. According to definition 2 and property 1.i), an entry $m_{i,j}$ from $\mathcal{M}$ is different from the empty set. Thus $\exists s_y \in S_{Ws}$ with at least one input $r_{i.label} \in \mathcal{T}$ and one output $Out\_s_y \in \mathcal{T}$ such that $Sim_{\mathcal{T}}(Out\_s_y, c_{j.label}) \neq 0$. Since $c_{j.label} \in In(s_x)$, two Web services s_x and s_y in S_{Ws} exist such that an output of the Web service s_y is consumed by an input of another Web service s_x. Thus s_x and s_y are sequence-composable.

□

By the Sequence Composability theorem of SLMs we proved that SLMs are able to not only store valid semantic links between Web services but also sequence composable Web services.

Example 16. ***(Sequence Composability in SLMs)***
Let $\mathcal{M}$ be the SLM illustrated in Example 15. S_b and S_a are sequence-composable in S_{Ws} iff $S_b \circ S_a$. Indeed there exists a pair $(i,j) = (5,6)$ in $\mathcal{M}$ such that $(r_{5.label}, c_{6.label}) = (PhoneNum, SlowNC)$. In the considered case $(S_a^-, 1) \subseteq m_{5,6}$ where $c_{6.label}$ is $SlowNC \in In(S_b) \subseteq \mathcal{T}$ and $r_{5.label}$ is $PhoneNum \in In(S_c) \subseteq \mathcal{T}$. Therefore an output of S_a is exploited by the input of S_b since $Sim_{\mathcal{T}}(Out\_S_a, In\_S_b) \neq 0$.

In case of more complex compositions, several services can be required to be chained with s_x in order to produce all input parameters of s_x e.g., Figure 3.6. So concurrency between some services needs to be considered. In case a service is in several entries of the same column, this means that concurrency can be required to compose with such a service.

Example 17. ***(Concurrency in SLMs)***
Let $\mathcal{M}$ be the SLM. The intersection of entries $m_{3,2}$ and $m_{5,2}$ is $\{(S_c, \frac{3}{4})\}$. This simply means that Web service S_c requires two input parameters $r_{3.label}$ i.e., $FastNC$ and $r_{5.label}$ i.e., $PhoneNum$ to be achieved. In other words a composition that requires the Web service S_c have to provide two output parameters which will be exploited by the input parameters of S_c.

Besides to model sequence composability and concurrency in Web service composition, SLM is also able to model the multiple choice. This case of non determinist compositions are conceivable in case an the number of elements in an entry of an SLM is strictly greater than one.

Example 18. ***(Non Deterministic Choice in SLMs)***
Let $\mathcal{M}$ be the SLM. The entry $m_{1,3}$ of $\mathcal{M}$ is $\{(S_a^-, \frac{1}{2}), (S_a, \frac{1}{2}), (S_a^+, 1)\}$. The latter set identifies that three Web services S_a, S_a^- and S_a^+ are able to provide an output parameter semantically close to the concept $FastNC$ i.e., $c_{3.label}$. In other words a composition that requires the input parameter $FastNC$ may choose between three different Web services.

According to the previous features of SLM, sequence, choice (aka Non Determinism) and concurrency of Web services can be retrieved and then used to model compositions returned by our approach (Requirement $R^{Composition}_{Expressivity}$).

3.3.3 Synthesis

Some Concluding Remarks

As studied in this Section, the SLM model aims at supporting not only trivial composition and basic composition such as *sequence composable* Web services but also more expressive compositions of Web services (Requirement $R^{Composition}_{Expressivity}$) such as non deterministic choice of services, or concurrent composition of services.

3.4 Conclusion

In this Chapter the **semantic link** concept as been introduced as the main composabilty criteria to achieve Web service composition. Towards this issue we focused on functional input and output parameters of services. The **requirement** $R^{Service}_{Applicability}$ is addressed by the industrial applicability of our model to OWL-S, WSMO, SWSO or SA-WSDL. The **requirement** $R^{Semantic}_{Composability}$ is addressed by the **semantic links**. The **requirement** $R^{Service}_{Expressivity}$ is addressed by the **functional description of input and output parameters**.

In addition we studied non standard matchmaking functions to direct semantic relations between services i.e., Intersection [124], Abduction [55], Concept Difference [198]. From this innovative valuation of semantic links between services, new issues related to robustness links have been identified.

From the definition of semantic link, a formal and flexible model (**SLM i.e., Semantic Link Matrix**) for Web service composition has been studied. The SLM model is used to pre-chain Web services according to the semantic similarity computed on semantic links (actually all valued possible interactions between all relevant Web services). This model facilitates composition of Web services by considering the latter computation achieved. We remarked that the required background for Web service composition at functional level i.e., valid semantic links and their semantic dependency can be known by means of SLMs of the domain. Indeed this model aims at supporting expressive compositions of Web services. On the one hand the **requirement** $R_{Flexibility}$ is addressed by the **SLM** model. On the other hand the **requirement** $R^{Composition}_{Expressivity}$ is addressed by the expressive compositions (i.e., sequential, non deterministic, concurrent) supported by the latter model.

Table 3.5 describes in details the requirements supported by the model introduced in Chapter 3.

By considering Web services described by means of their input and output parameters, Web service composition can be studied as a *semantic links composition*.

Chapter 4 suggests two approaches that apply AI planning techniques to computing a partial ordering of Web services arranged in a simpler version of a workflow that fulfils a composition goal and Requirement $R^{Composition}_{Expressivity}$.

According to the latter formal model together with its semantic links, the Chapter 4 first presents a composition approach, which is mainly oriented by i) the SLM of a given domain, and ii) its valid semantic links together with their values since all Web services are semantically well ordered in the robust SLM model. The resulting ordering will be i) composed of Web services, ii) semantic dependences i.e., semantic links [114] and iii) as robust as possible (Definition 3.1.4 in Section 3.1). Requirement $R^{Service}_{Expressivity}$ is supported in part (input and output parameters), and non determinism is focused on Web services (i.e., non deterministic choice of service to perform a task) and not on their parameters.

In addition the Chapter 4 studied another approach to compute conditional compositions of Web services. Here Web services are described by means of their input, output parameters, together with their preconditions and effects (full part of $R^{Service}_{Expressivity}$). The conditional compositions will be computed by means of i) their semantic dependences and ii) the causal laws

<table>
<tr><th rowspan="2">Requirement R_i</th><th rowspan="2">Details</th><th colspan="3">Chapter 3</th></tr>
<tr><th>Section 3.1</th><th>Section 3.2</th><th>Section 3.3</th></tr>
<tr><td rowspan="2">$R^{Composition}_{Automation}$</td><td>Formalism</td><td colspan="3" rowspan="2">✗</td></tr>
<tr><td>Composition Mechanism</td></tr>
<tr><td rowspan="2">$R_{Expressivity}$</td><td>$R^{Service}_{Expressivity}$</td><td colspan="3">In this Chapter, two main functional parameters have been studied to describe Web services i.e., Inputs and Outputs, annotated by concepts using a common domain ontology. Information-providing services.</td></tr>
<tr><td>$R^{Composition}_{Expressivity}$</td><td colspan="2">✗</td><td>The SLM model supports sequence composability of services, non determinism choice of services and concurrent compositions.</td></tr>
<tr><td rowspan="2">$R_{Composability}$</td><td>$R^{Semantic}_{Composability}$</td><td colspan="3">Semantic dependence through (valid and robust) semantic links. Valuation with basic matchmaking functions i.e., Exact, PlugIn, Subsume, Disjoint and extra functions i.e., Intersection, Abduction and Difference to compute robust semantic links. Computation at Design Time.</td></tr>
<tr><td>$R^{Causal}_{Composability}$</td><td colspan="3">✗</td></tr>
<tr><td colspan="2">$R_{Flexibility}$</td><td>✗</td><td colspan="2">Supported by the SLM model.</td></tr>
<tr><td colspan="2">$R_{Optimization}$</td><td colspan="3">✗</td></tr>
<tr><td rowspan="2">$R_{Applicability}$</td><td>$R^{Service}_{Applicability}$</td><td colspan="3">Applicable to the OWL-S service profile, WSMO service capability, SWSO Inputs/Outputs or SA-WSDL (see Section 1.3.2 for further details).</td></tr>
<tr><td>$R^{Composition}_{Applicability}$</td><td colspan="3">✗</td></tr>
</table>

Table 3.5: Table of Requirements supported by Chapter 3. Legend: ✗ = not addressed.

($R^{Causal}_{Composability}$, see Section 2.1.1 for further details). By studying conditional compositions non determinism of output parameters of Web services will be considered.

Chapter 4

Semantic Link and Causal Law based Composition

In this chapter, we describe two complementary approaches for web Service composition. Both approaches focus on **requirement** $R^{Composition}_{Automation}$ to achieve composition.

- The first approach focuses only on semantic links between output and input parameters of Web services as composabilty criteria. Preconditions of all Web services involved in a composition are supposed true before execution (e.g., satisfied by some effects of some preceding Web services in the composition, or true in the initial situation). Towards this issue we suggest to apply AI planning techniques on a flexible model i.e., the SLM (Semantic Link Matrix in Chapter 3) of a given domain. The result is a partial ordering of Web services arranged in a simpler version of a workflow that fulfils a given composition goal. This ordering will be composed of i) Web services and ii) their semantic links (as robust as possible - See Definition 13).

  In this approach a part of **requirement** $R^{Service}_{Expressivity}$ is ensured since only input and output parameters are required. The **requirement** $R^{Semantic}_{Composability}$ is supported since semantics links are considered as composability criteria. Finally the control constructs supported by the formal SLM model satisfies the **requirement** $R^{Composition}_{Expressivity}$.

Even if the first approach is quite appropriate in some scenarios (e.g., Web services description using SA-WSDL), there are some limitation. For instance we cannot restrict Web services to be described by only their functional input and output parameters. Indeed some industrial applications may require to compose Web services by means of their input, output parameters (e.g., SA-WSDL based Web service description), and of relationships between services, preconditions and effects (e.g., WSMO or OWL-S based Web service description). This is a related to the open world of Web services.

- The second approach does not longer consider assumption defined in Section 4.1.1 (assumption that any precondition of any service is automatically true in any time of the composition). Towards this issue, semantic links between output and input parameters together with causal laws (i.e., *causality relationships* between effects and preconditions of services and complex, explicit relationships between services - see Section 2.1.1 and Figure 2.1) are considered and integrated to compute compositions of Web services in this

approach. Information-providing or/and world-altering services can be supported in this approach. Section 4.2 presents an augmented and adapted version of the logic programming language Golog [121] i.e., s_{sl}Golog as a natural formalism not only for reasoning about the latter links and laws, but also for automatically composing services. s_{sl}Golog operates as an offline interpreter that supports conditional compositions of services.

In this approach the **requirement** $R^{Service}_{Expressivity}$ is supported by a high level of Web service description i.e., input, output parameters and preconditions, effects. Both the **requirement** $R^{Semantic}_{Composability}$ and $R^{Causal}_{Composability}$ are considered as composability criteria. Finally the **requirement** $R^{Composition}_{Expressivity}$ is supported by considering sequential and conditional compositions.

The main distinctions between these two approaches is about **requirements** $R^{Service}_{Expressivity}$, $R_{Composability}$ and $R^{Composition}_{Expressivity}$.

The remainder of this Chapter is as follows. Section 4.1 presents the first approach whereas Section 4.2 presents the second approach. Finally Section 4.3 draws some concluding remarks.

4.1 Semantic Link based Web Service Composition

In this section we focus on a automated (requirement $R^{Composition}_{Automation}$) semantic link based Web service composition (requirement $R^{Semantic}_{Composability}$). Web services are described by means of their input and output parameters (a part of requirement $R^{Service}_{Expressivity}$). More formally we suggest to apply AI planning techniques on a flexible model i.e., the SLM of a given domain. The latter model ensures to obtain expressive composition of Web services (requirement $R^{Composition}_{Expressivity}$).

The remainder of this Chapter is as follows. First of all, Section 4.1.1 states about the status of causal laws in this composition approach. Section 4.1.2 revisits Web service composition as an AI planning problem. In Section 4.1.3 we formalize compositions constructs to efficiently model Web service composition. Section 4.1.4 describes in details the composition process. In Section 4.1.5 we give some main properties of the computed compositions. Section 4.1.6 focuses on robustness in Web service composition. Finally Section 4.1.7 gives some limitations of the model and concludes.

4.1.1 Disregarding Causal Laws between Web Services

As previously said in Section 2.1, causal laws can be used as a composability criterion to compute Web service composition. On the one hand the causal laws ensure that preconditions of services are satisfied before any Web service execution (i.e., *causality relationships*). On the other hand they add more explicit and complex relationships between parameters (under some conditions) of different Web services than semantic links.

In this approach we assume for sake of simplicity that:

- *causality relationships* are trivial i.e., *preconditions of all Web services involved in a composition are true* (e.g., a priori, satisfied by effects of other services or in the initial situation) in the composition;

- *complex relationships between services* (e.g., relationships that link an input parameter of a service to an output parameter of another service under some conditions) are disregarded.

Disregarding open issues related to preconditions, effects and causal laws in this work, first does simplify the composition process, second enables us to highlight open issues related to semantic composability (and its data flow) of Web services such as robustness, and third allows scaling up to large sets of capabilities; in turn, a relevant part of the interaction taking place amongst services is ignored.

4.1.2 Web Service Composition as a Revisited AI Planning Problem

From an SLM of a given domain with its valid semantic links, together with basic AI planning techniques such as regression, or progression-based search we aims at computing complete, correct, consistent and robust plans as solutions of a composition problem.

The method consists in computing a (or some) composition(s) of services that produces instances of the desired concepts in $\beta \subseteq \mathcal{T}$ depending on some individuals in an ABox $\mathcal{A}$ (e.g., Figure 4.1) of the ontology T.

To this end, we suggest to compute a (or more) solution of the AI planning-based Web service composition problem (Definition 18).

Definition 18. ***(AI Planning-based Web Service Composition Problem)*** *An AI planning-based Web service composition problem [54, 144, 167, 174] is defined as a triple* $\Pi = \langle S_{Ws}, \mathcal{A}, \beta \rangle$. *The set of Web services* S_{Ws} *refers to planning operators (i.e., available actions),* $\mathcal{A}$ *and* β *are respectively an adaptation of the initial states and goal of the AI planning problem.*

The main differences with AI planning problem concern first the description of goals and Initial states, second addition of assumptions on the planning operators, the composition *goal* and initial conditions. The latter assumptions are required to stage with open issues related to the AI planning-based Web service composition [195].

In this direction the set of Web services S_{Ws} is closed by assumption. Moreover the output parameters of Web services are not consumed, and then can be re-used by several different Web services in the composition.

Unlike goals which are described as fluent with a first-order logic representation in AI planning domain, the goal β is clearly described as a set of defined concepts in a TBox. β informs about composition (or plan) directions in order to retrieve some individuals of concepts in β. The DL-based description of β allows to make reasoning such as satisfiability, subsumption, abduction, and concept difference especially to infer some properties between concepts in β and parameters of Web services in $\mathcal{T}$.

The Initial state of our problem is modelled as a set of instances in an ABox $\mathcal{A}$. In other words $\mathcal{A}$ informs about initial conditions by defining some instances of concepts we can use to instantiate input parameters of some Web services.

Then a composition of Web services is computed in a well-defined domain: goals are explicitly given, initial state is well defined and Web services are strictly defined at functional level. Therefore non determinism, implicit goals, fuzzy Web service descriptions and behaviours are not

considered in our composition approach. It does seem possible to directly apply (some) current AI planning methods to our specific problem (see Chapter 2).

$\mathcal{A}$-I *freddy.lecue@orange-ftgroup.com* : **Email**

$\mathcal{A}$-II *+33299124625* : **PhoneNum**

$\mathcal{A}$-III *35512* : **ZipCode**

Figure 4.1: Sample of the Assertional Box of the $\mathcal{ALE}$ Domain Ontology T.

4.1.3 Modelling Composition as a Partial Ordering of Web Services

Before mapping a Web service composition as a simpler form of an AI planning problem, we suggest to briefly overview the way we model a composition of Web services. Considering S_{Ws} be the set of available and relevant services, the composition result is a partial ordering $(\circ,\ S_{Ws})$ of services in S_{Ws} arranged in a simpler workflow (mainly sequence, concurrency and non determinism operators - Requirement $R^{Composition}_{Expressivity}$) in order to fulfil the composition goal β. The latter ordering of Web services is intertwined with valid semantic links. As emphasized by Chapter 3 *a composition is then defined as a plan of services wherein all services are semantically well ordered and well linked by semantic links.*

Example 19. ***(Semantic links and Ordering on Web Services)*** *Suppose S_a and S_b be two Web services illustrated in Example 4 and Table 3.3. Since S_b and S_a are sequence composable i.e., $S_b \circ S_a$, S_a required to be executed first in order to provide the input parameter required by S_b.*

The partial ordering $(\circ,\ S_{Ws})$ returned by our FLC approach consisted of Web services together with some characteristic elements i.e., *predecessors* and *successors*. In a nutshell a Web service s_x is a predecessor of another Web service s_y i.e., $s_y \circ s_x$ in case s_x provides at least one output parameter to s_y. Reciprocally s_y is a successor of s_x.

Since the order is not necessarily a total order some Web services in $(\circ,\ S_{Ws})$ can be incomparable i.e., two Web services cannot be ordered since none of both requires the achievement of the other service. In this direction concurrency of Web services s_x and s_y are conceivable in service composition in case i) s_x and s_y are incomparable and ii) their output parameters are used by input parameters of a same service. Therefore the partial ordering of Web services models not only sequences of comparable Web services but also the concurrent execution of incomparable Web services.

The following definition is introduced to model a partial order of Web services by means of sequence, non determinist choice and concurrency[1].

Definition 19. ***(Main Constructs for Web service composition)*** *Let $\langle S_{Ws}, \mathcal{A}, \beta\rangle$ be the AI planning-based Web service composition problem, constructs of Web service composition and their priority order are $\wedge > \vee > \circ$, such that:*

- *$\wedge$ is the conjunction operator i.e., the concurrency construct (or AND-Branching);*
- *$\vee$ is the disjunction operator i.e., the non determinist choice construct;*

[1]Such constructs are supported by the SLM model introduced in Chapter 3.

- *$s_i \circ s_j$ refers to the sequence construct. Such a construct models sequence-composability (Definition 17). Therefore $s_i \circ s_j$ is possible only if $\exists C_o, C_i \in \mathcal{T}$ such that $\langle s_j, Sim_{\mathcal{T}}(C_o, C_i), s_i \rangle$ is a valid semantic link.*

Example 20. ***(Semantic Links and Partial Ordering on Web Services)***
Let $S_{y_1}, S_{y_2}, S_{y_3}$ and S_x be the four Web services illustrated in Figure 3.6, the partial order $(\circ, \{S_{y_1}, S_{y_2}, S_{y_3}, S_x\})$ is defined by

(i) $S_x \circ S_{y_1}$;

(ii) and $S_x \circ (S_{y_2} \vee S_{y_3})$.

The reduced form is $S_x \circ (S_{y_1} \wedge (S_{y_2} \vee S_{y_3}))$.

4.1.4 A Regression-based Approach for Web Service Composition (Ra_4C)

In this section we focus on a AI planning based search. More specifically we apply a Regression-based Approach for Composition i.e., Ra_4C. The suggested composition process is presented and detailed in Algorithm 2.

In a nutshell the composition process follows a recursive and regression-based search from the set of concepts in β (concepts that do not have any individuals in the ABox $\mathcal{A}$) with initial conditions in $\mathcal{A}$.

The main idea consists in controlling and parsing the SLM of a given domain in an adequate way to obtain a composition of services and their semantic links that will satisfy the goal composition. The latter goal is satisfied in case the composition candidate retrieves (one or more) individual(s) of all concepts in the goal β.

The Ra_4C approach requires a first call of the Algorithm 2 as follows $Ra_4C(\mathcal{M}, \emptyset, \langle S_{Ws}, \mathcal{A}, \beta \rangle, \emptyset)$. This step is required to compute automated Web service compositions that provide (one or more) individual(s) of concepts in the goal β. In the trivial case (line 7) wherein all concepts in β have already individuals in $\mathcal{A}$, the goal β is obviously fulfilled by the initial condition. The automated process of composition is then stopped since trivially satisfied by $\mathcal{A}$. In the more complex case (line 10) wherein the ABox $\mathcal{A}$ does not contains any individuals of the concept β_i in β, the composition process needs to retrieve at least one Web service s_x in S_{Ws} with β_i defined as one of its output parameters. In other words a Web service discovery process eased by the SLM of the domain is performed. In case of a discovery success (line 14), the process is iterated with the s_x input parameters as new goals (line 23) of AI planning-based Web service composition. Alternatively (line 26), the process is stopped and the (or a part of the) plan is reduced to an incorrect composition $\emptyset$ since the current goal is considered as open i.e., no composition or individuals in $\mathcal{A}$ can solve the current goal. All the previous process is executed recursively (line 25) until the concepts in β and new goals (recursive goals as input parameters of services involved in the composition) have individuals in $\mathcal{A}$ (stop condition in line 7). Therefore the Algorithm 2 returns a disjunction of consistent plans consisted of valid and *sequence-composable* semantic links i.e., a (or more) composition(s) of Web services semantically chained by semantic links.

Computational Complexity of Ra_4C

Formally the computational complexity of the Ra_4C process is polynomial with the number of Web services and their numbers of functional (i.e., here input since we perform a regression-based search) parameters. The latter complexity is obviously related and very depending on the filling rate of the SLM. It is obvious that the sparser the SLM the faster the Ra_4C process is.

Algorithm 2: Regression-based Approach for Composition Ra_4C.

Input: A SLM $\mathcal{M}$ ($[m_{i,j}]$), a (or disjunction of) plan(s) π, an AI planning based Web service composition problem $\langle S_{Ws}, \mathcal{A}, \beta \rangle$, a set of solved goals β_{sv}, a set of non valid goals β_{nv}.
Result: A disjunction of consistent plans π.
begin
  // Temporary set of pairs in $S_{Ws} \times (0, 1]$.
  $S_c \leftarrow \emptyset$;
  // Stop condition of the Ra_4C algorithm.
  if $\exists c_k \in \mathcal{A}$ *such that* ((c_k *is an individual of* C_k) & ($Sim_{\mathcal{T}}(C_k, \beta) \neq 0$)) **then**
    $\pi \leftarrow \beta$;
  // Web services discovery with β output.
  foreach $I_i \in Input(S_{Ws})$ **do**
    if $\exists (s_y, v) \in m_{I_i,\beta}$ **then**
      Add($(s_y, v), S_c$);
  // Plan for Web service composition.
  if $S_c \neq \emptyset$ **then**
    foreach *pair* $(s_y, v) \in S_c$ *such that* $s_y \in S_{Ws}$ **do**
      $\pi \leftarrow \pi \vee s_y$;
      foreach $In\_s_y \in In(s_y)$ **do**
        if $\beta \in \beta_{sv}$ **then**
          // Case of Inconsistent plan. Detection of Loops.
          $\pi \leftarrow \pi \wedge \emptyset$;
          Add(β, β_{nv});
        else
          Add(β, β_{sv});
          $\Pi \leftarrow \langle S_{Ws}, \mathcal{A}, In\_s_y \rangle$;
          $\pi \leftarrow \pi \circ (\bigwedge_{In(s_y)} Ra_4C(\mathcal{M},\pi,\Pi,\beta_{sv}))$;
  else
    // Non correct plan since there is an open goal (input).
    $\pi \leftarrow \pi \wedge \emptyset$;
  return π;
end

Experiment results of this approach are presented in Chapter 7.

In the Ra_4C process we assumed without loss of generality, and for sake of simplicity that β refers a unique concept we want to instantiate, and not as a set of concepts. Even if we do not investigated on a subtle method to perform a composition with multi goals, the practical computation of a composition with n concepts in β is linear with n. However it could be even less in general case e.g., some goals can be satisfied by a composition of services satisfying another goal.

4.1.5 Properties of Web Service Compositions Computed with Ra_4C

In the same direction as pure AI planning problem and their solutions (aka., plan), the set of computed compositions by the Algorithm 2 has some interesting properties [75] related to their consistency, completeness and correctness. Such properties are depending on i) the set of Web services S_{Ws} involved, ii) the set of concepts β as goal and iii) the ABox $\mathcal{A}$ of the given domain. In the following we briefly describe the properties related to these compositions of Web services.

Consistency

The consistency property in Web service composition is a necessary condition to obtain executable solutions. This condition is satisfied by compositions containing no cycle in the ordering constraints and no semantic link conflicts [178]. Such compositions are computed by means of the Ra_4C algorithm. Indeed the latter algorithm identifies cycles and conflicts to dispose of inconsistent semantic links. This inconsistency is tackled by a simple update of solved goals (line 23). Therefore a goal is not resolved twice or more times during the composition process.

Example 21. ***(Set of consistent compositions)***
Let $\mathcal{M}$ be the SLM illustrated in Example 15, and $\Pi = \langle\{S_a-, S_a, S_a^+, S_b, S_c, S_d\}, \mathcal{A}, \{Invoice\}\rangle$ be the planning-oriented service composition problem we plan to solve. In a nutshell the goal of the composition is to compute a partial ordering of services that enable us to provide an `Invoice`*, given some elements in the ABox and some available and relevant Web services. More specifically, by considering $\mathcal{A}$ illustrated in Figure 4.1, the composition goal is to obtain an Invoice of a customized service. In a previous step the end user has selected a set of offers (i.e., services) she wants to subscribe, together with some information such as her email address, Phone number and the ZipCode of the desired phone line. From this the composition result of Ra_4C is a disjunction of nine consistent compositions (or simpler plans) depicted in Figure 4.2:*

$$\begin{aligned}\pi_{S_x,S_y} := S_d \circ \Big[& PhoneNum \\ & \wedge (S_c \circ (S_x(Email, PhoneNum, ZipCode) \wedge PhoneNum)) \\ & \wedge (S_b \circ (S_y(Email, PhoneNum, ZipCode) \wedge PhoneNum))\Big]\end{aligned}$$

wherein S_x and S_y can be any Web service in $\{S_a, S_a^-, S_a^+\}$.

Correctness

Adapted from the AI planning research area [178], we define a correct composition as a composition wherein every input of every Web service can be provided by an output parameter of another Web service or by an individual occurring in the ABox $\mathcal{A}$. In other words correctness of resulting compositions is guaranteed in case they do not contain any open inputs. In our approach, all non correct compositions are identified in the Ra_4C process in line 26 of algorithm 2. Therefore compositions with open inputs are removed from the set of potential solutions.

Example 22. ***(Set of correct compositions)***
The nine composition results returned by Algorithm 2 are correct since none of them contains any open goals and inputs.

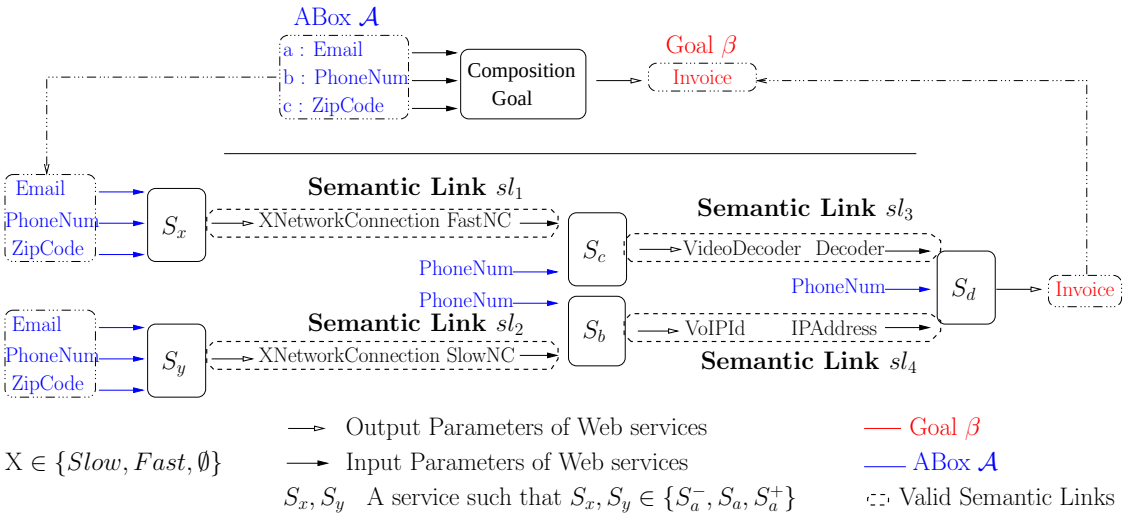


Figure 4.2: Ra_4C Result on the Motivating example.

Completeness

Here we study the completeness properties of compositions computed by Algorithm 2.

By definition, **an SLM contains all required information about complete plans since an SLM explicitly stores all valid semantic links** between sequence composable services (see Section 3.3).

According to the Ra_4C process the compositions refinement follows a backward chaining strategy from the goal β to initial states $\mathcal{A}$ by means of a domain SLM and its semantic links. Therefore by definition of SLM, all compositions of services consist of valid semantic links $\langle s_y, Sim_{\mathcal{T}}(Out\_s_y, \beta), s_x \rangle$ hence complete compositions.

Example 23. ***(Set of complete compositions)***
The set of compositions returned by Ra_4C is complete by definition.

By means of the Ra_4C process a set of correct, complete and consistent compositions is returned. However the Algorithm Ra_4C does not study how to overcome the issue of robustness in Web service composition. This issue is addressed in the next Section.

4.1.6 Robust Semantic Web Service Composition

Even if Ra_4C is able to compute correct, complete and consistent compositions of Web services, some of them can be not robust enough. Indeed some compositions returned by Ra_4C can contain non robust semantic links (Definition 13) hence non robust compositions (Definition 14).

Example 24. ***(Non Robust Composition of Web Services)***
Suppose π_{S_a,S_a} (from π_{S_x,S_y} in Example 21) be a composition result illustrated in Figure 4.3. Its

formalization is as follows:

$$\pi_{S_a,S_a} := S_d \circ \Big[PhoneNum \wedge (S_c \circ (S_a(Email, PhoneNum, ZipCode) \wedge PhoneNum)) \wedge (S_b \circ (S_a(Email, PhoneNum, ZipCode) \wedge PhoneNum))\Big]$$

Such a composition of Web services is not robust since three of its four semantic links are not robust i.e., sl_1, sl_2 and sl_4.

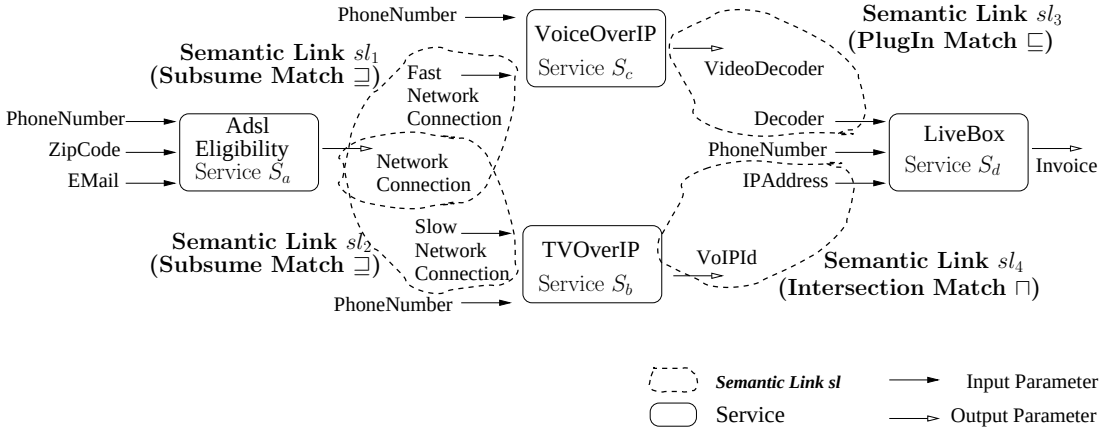


Figure 4.3: A Non Robust Composition of the Motivating example.

In the considered case a complete automation of semantic Web service composition is still not a reality, especially when a Web service composition comprises non robust semantic links.

An intuitive but naive method would be to not consider non robust semantic links i.e., semantic links valued by a Intersection and Subsume match level. Therefore the composition approach would be the same as proposed by the Ra_4C process with a restricted SLM in $\mathrm{M}_{p,q}(\mathcal{P}(S_{Ws} \times \{1, \frac{3}{4}\}))$. It is obvious that such a method is far from convenient since it would consider no more than two matchmaking levels to value semantic links, hence a loss of expressivity in semantic links.

Another approach consists in replacing non robust semantic links of a composition with their robust forms as suggested in Section 3.1.5. Therefore mismatched connections between Web services can be repaired through a fine-grained process of mediation (actually finer that the WSMO mediation). The Web service composition process is still automatic in case the *Extra Description* required by the non robust semantic links is automatically computed.

In the following we study two main methods to obtain this *Extra Description* i.e.,

- the first method performs robust Web service composition in an automated way. This case is denoted as the *Perfect Case*.
- the second method acts in a semi-automated way by *relaxing constraints* to the end-user.

The Perfect Case

An intuitive method to immediately compute the *Extra Description* consists in *discovering services* that return this description by means of their output parameters.

To this end, the *Extra Description* of non robust semantic links is first computed according to Concept Difference (or Concept Abduction).

The computed *Extra Description* is then exposed as a goal of a Web service discovery process. The latter process will be in charge of discovering relevant Web services. Such services will be able to provide the *Extra Description* as output parameters.

It is obvious that the *Extra Description* can be reached by one or a conjunction of Web services, depending on the *Extra Description* and the discovery process.

The main constraint of this method is related to the computational complexity of the process. Indeed all input parameters of new discovered Web services have to be considered as a new composition goal. Such goals have to be either known at run time or linked to an output parameter of another Web service through a robust semantic link in the final composition. The latter consideration limits the scalability of the approach in case i) the discovery process is time consuming and ii) the number of discovered services is large.

In this approach, the more non robust semantic links in a composition the more large the number of services in this composition. Such a solution can be employed and implemented in any composition approach, but its computational complexity is its main drawback.

An Alternative Approach: Relaxing Constraints

In case wherein no service can reach the *Extra Description* there is no way to automatically compute this description. In this direction all available information does not guarantee to find a robust Web service composition.

Consequently, the latter description has to be computed *by relaxing some constraints* during the composition process.

Indeed relaxing some constraints and obtaining a composition of robust semantic links is an interesting trade-off to reach composition. These constraints still guarantee the original feasible solutions and yield additional feasible solutions. Constraints $B_{i,1\leq i\leq n}$ in Web service composition refer to the *Extra Description*. More formally the set of relaxing constraints $\mathcal{B}$ is expressed by Definition 20 where $\langle s_y, Sim_{\mathcal{T}}(Out\_s_y, In\_s_x), s_x\rangle$ refers to semantic links in the composition model.

Definition 20. ***(Set of Relaxing Constraints)***
The set of relaxing constraints $\mathcal{B}$ is defined by

$$\inf_{\sqsubseteq}\{In\_s_x \backslash Out\_s_y | \langle s_y, Sim_{\mathcal{T}}(Out\_s_y, In\_s_x), s_x\rangle \textit{ is a valid semantic link}\}\backslash_{set} \{\top\} \quad (4.1)$$

Intuitively, the set of relaxing constraints $\mathcal{B}$ of a Web service composition is defined as being the set of descriptions able to change non robust semantic links into their robust forms. $\mathcal{B}$ gathers the most specific descriptions of the set $\{In\_s_x \backslash Out\_s_y\}$ where $\langle s_y, Sim_{\mathcal{T}}(Out\_s_y, In\_s_x), s_x\rangle$ is a valid semantic link. The latter consideration implies that a same description (i.e., the most specific) can be used by a finite set of non robust semantic links to change them into their robust forms. The descriptions used to perform these changes will be the most specific descriptions of the *Extra Description*. $\mathcal{B}$ does not only explain why the composition process failed but also gives a solution of the robustness problem of semantic links hence a way to reach robust Web service composition.

Proposition 2. ***(Constraints for Robust Semantic Links)***
The set of relaxing constraints $\mathcal{B}$ of a Web service composition with only robust semantic links is the empty set.

Proof. Let π be a service composition constituted of only robust semantic links $sl_{i,1\leq i\leq n}$, defined by $\langle s_y, Sim_{\mathcal{T}}(Out\_s_y, In\_s_x), s_x\rangle_i$. By definition, the match level between Out_s_y and In_s_x is either Exact or PlugIn, hence $Out\_s_y \equiv In\_s_x$ or $Out\_s_y \sqsubseteq In\_s_x$. By difference (3.1) we obtain in the two cases that $In\_s_x \backslash Out\_s_y \equiv \top$ i.e., $\mathcal{B}$ is defined by the empty set. □

Once the set of *Extra Descriptions* is computed through Concept Difference, the set of relaxing constraints $\mathcal{B}$ (Definition 20) is computed to be suggested to the end user in order to be relaxed. This user is then responsible to provide the *Extra Description* required by the system in order to elaborate the final and robust Web service composition, hence satisfying the initial user request. The suggested method has the advantage of relaxing constraints on the end user side.

In the motivating example, Web services and user's requirements are both not specified enough to focus on the *Extra Description* and also advantages of relaxing constraints. The relaxing task is of the utmost importance in real scenarios of composition since Web service composition often requires some refinements such as relaxing constraints to turn into an automated composition.

Example 25. ***(Relaxed Semantic links)***
The motivating example exposes a Web service composition through 4 semantic links $sl_{i,1\leq i\leq 4}$ (Figure 4.3). Three of the four valid semantic links are not robust. Indeed

- *the semantic links $sl_{i,1\leq i\leq 2}$ are valued by a Subsume match;*
- *the semantic link sl_3 is valued by an Intersection. match.*

No Web service may provide the Extra Description necessary to form robust semantic links. A Relaxing constraints needs to be applied to obtain a composition of robust semantic links. The discovery of the Extra Descriptions $\mathcal{B}$ gives directions to obtain robust semantic links.

According to the Definition 20, $\mathcal{B}$ is constituted of an union of three differences in DL i.e., the difference between the concepts

i) FastNetworkConnection and NetworkConnection to change sl_1 by a semantic link valued by an Exact match;

ii) SlowNetworkConnection and NetworkConnection to change sl_2 in the same way as sl_1;

iii) IPAddress and VoIPId to replace sl_3 by a semantic link valued by a PlugIn match.

Since sl_4 is a robust semantic link and $\forall netSpeed.AdslMax \sqsubset \forall netSpeed.Adsl1M$, $\mathcal{B}$ is defined by $\{\forall netSpeed.AdslMax, \forall protocol.IP\}$.

The Web service composition can be automatically computed in case the *Extra Description* is provided by the end user, other services or any third party, depending on the application we want to automate. For instance $\forall netSpeed.AdslMax$, $\forall protocol.IP$ can be provided by another Web service in case we want to automate the composition process of the motivating example.

An Approach for Robust Semantic Web Service Composition: Robust Ra_4C

The composition approach used and extended in this Section is **based on the model Ra_4C introduced in Section 4.1.4, but can be easily applied with many other approaches of functional level composition** e.g., [210, 105, 190].

From i) some user constraints, ii) a set of Web services and iii) a goal to achieve the Ra_4C approach computes a composition consisted of valid but not necessarily robust semantic links. Indeed some solution proposals may refer to non robust compositions since some semantic links can be valued by a Subsume match level. It is obvious that this approach and most of the functional level composition methods need refinements to perform robust Web service composition. The algorithm suggested in this section aims at extending not only the Ra_4C approach but also any other functional level composition method in order to overcome the robustness problem in Web service composition.

The main idea is as follows: the set of relaxing constraints $\mathcal{B}$ is progressively evaluated throughout the computation of composition by Ra_4C. This method is suitable especially to compare different descriptions and then rapidly prune the worst solutions.

Algorithm 3 differs from Ra_4C and other composition approaches, primarily because it does explore non robust semantic links and stop a composition process in case its *Extra Description* is more specific than one of the pre-computed solutions. The more specific *Extra Description* the more description have to be provided to enable automation of the composition. That is why the best Web service compositions are supposed to be compositions with the most general *Extra Descriptions*, i.e., with the least constraints.

The Algorithm 3 consists of the following steps. First (line 4 and 5) in the trivial case wherein the algorithm Ra_4C returns a robust composition, the process is stopped. In the more complex case (from line 7), each composition result (line 9) is analysed by algorithm 3. During the computation of each composition result π_i (line 9), the *Extra Description* of each non robust semantic link involved in π_i is computed. By assuming $\mathcal{B}$ be the *Extra Description* of the best current robust composition, we compare the current *Extra Description* $\mathcal{B}_{\pi_i}$ and the best current *Extra Description* $\mathcal{B}$ (lines 12 and 14). In case $\mathcal{B}_{\pi_i}$ (line 12) is more specific than $\mathcal{B}$, this means that $\mathcal{B}$ is the most appropriate *Extra Description*. The more the composition process proceeds the more specific will be $\mathcal{B}_{\pi_i}$. Indeed more non robust semantic links can be involved in the composition π_i. That is why we stop the process of computing the *Extra Description* of π_i. Otherwise in case $\mathcal{B}_{\pi_i}$ is more general than $\mathcal{B}$ (line 14) we continue the process of valuating the *Extra Description* of $\mathcal{B}_{\pi_i}$. At the end of the algorithm 3 (line 18), the best robust Web service composition together with its *Extra Description* are returned. Roughly speaking the set $\mathcal{B}_\pi$

required by non robust semantic links of the composition π is returned to the end user.

Algorithm 3: Robust Regression based Approach for Composition (Robust Ra_4C).

```
1  Input: A composition process Ra4C.
2  Result: The best compositions and their Extra Descriptions.
3  begin
4      if Ra4C returns a robust composition π_i then
5          return {(π_i, ∅)};
6      else
7          B_π0 ← compute Extra Description of π_0;
8          sol ← {(π_0, B_π0)};
9          foreach π_{i_{i≠0}} do
10             while π_i computation is in progess by Ra4C do
11                 B_πi ← current Extra Description of π_i;
12                 if B_πi ⊏ sol.B then
13                     stop π_i computation;
14                 if B_πi ⊒ sol.B then
15                     continue to build π_i;
16             if π_i is valid then
17                 sol ← sol ⊔ {(π_i, B_πi)};
18         return sol;
19 end
```

According to the set of available Web services i.e., commercial offers pre-selected by the end user, all semantic links are computed first and then Concept Difference reasoning is applied to non robust semantic links.

The computational complexity of the introduced method is the same as approaches without relaxation (e.g., Ra_4C) since the Concepts Differences are solved in a pre-processing phase for each non robust semantic links we consider in the composition. Therefore the computational complexity of the pre-processing step is mainly depending on the match levels of non robust semantic links, and particularly on the computational complexity of all Concepts Differences.

In case the Difference is processed at run time, the overall algorithm would no longer be a polynomial time algorithm (with an oracle for subsumption), especially due to the additional complexity caused by the computation of the difference.

Example 26. ***(Computing the Best Robust Composition)***
According to the Algorithms 2 and 3, the different Extra Description *of the nine compositions* $\mathcal{B}_{\pi_{S_x,S_y}}$ *can be computed with* $S_x, S_y \in \{S_a^-, S_a, S_a^+\}$. *The results are depicted in Table 4.1. According to this Table, it is obvious that the two best robust compositions are* $\pi_{S_a^+,S_a^+}$ *and* $\pi_{S_a^+,S_a^-}$ *since they have the most general* Extra Description. *However the latter compositions are not robust enough and require some* Extra Description $\{\forall protocol.IP\}$.

π_{S_x,S_y}	Semantic Link Value				$\mathcal{B}_{\pi_{S_x,S_y}}$
	sl_1	sl_2	sl_3	sl_4	
π_{S_a,S_a}	$\sqsupseteq$	$\sqsupseteq$	$\sqsubseteq$	$\sqcap$	$\{\forall netSpeed.AdslMax,\ \forall protocol.IP\}$
π_{S_a,S_a^-}	$\sqsupseteq$	$\equiv$	$\sqsubseteq$	$\sqcap$	$\{\forall netSpeed.AdslMax,\ \forall protocol.IP\}$
π_{S_a,S_a^+}	$\sqsupseteq$	$\sqsubseteq$	$\sqsubseteq$	$\sqcap$	$\{\forall netSpeed.AdslMax,\ \forall protocol.IP\}$
$\pi_{S_a^-,S_a}$	$\sqsupseteq$	$\sqsupseteq$	$\sqsubseteq$	$\sqcap$	$\{\forall netSpeed.AdslMax,\ \forall protocol.IP\}$
$\pi_{S_a^-,S_a^-}$	$\sqsupseteq$	$\equiv$	$\sqsubseteq$	$\sqcap$	$\{\forall netSpeed.AdslMax,\ \forall protocol.IP\}$
$\pi_{S_a^-,S_a^+}$	$\sqsupseteq$	$\sqsubseteq$	$\sqsubseteq$	$\sqcap$	$\{\forall netSpeed.AdslMax,\ \forall protocol.IP\}$
$\pi_{S_a^+,S_a}$	$\equiv$	$\sqsupseteq$	$\sqsubseteq$	$\sqcap$	$\{\forall netSpeed.Adsl1M,\ \forall protocol.IP\}$
$\pi_{S_a^+,S_a^-}$	$\equiv$	$\equiv$	$\sqsubseteq$	$\sqcap$	$\{\forall protocol.IP\}$
$\pi_{S_a^+,S_a^+}$	$\equiv$	$\sqsubseteq$	$\sqsubseteq$	$\sqcap$	$\{\forall protocol.IP\}$

Table 4.1: *Extra Description* $\mathcal{B}_{\pi_{(S_x,S_y)}}$ of the Motivating Example.

4.1.7 Synthesis

Some Limitations of the Ra_4C Approach

In computing the SLM, a single output from one service can be related to only a single input to a possible following service (see Definition 10 of semantic link). Since our approach is SLM oriented, this is a necessary restriction to compute first correct, consistent and complete compositions and then robust compositions.

It appears that our use of Description Logics for describing inputs and outputs would allow more generalizations of inputs and outputs to be matched. For instance a conjunction of output parameters from different Web services could be semantically matched to one (or more) input parameter(s) of a (or more) services. To this end the semantic link definition requires to be extended. Even if such an extension is straightforward, its SLM adaptation is more complex and even inappropriate for Web service composition. Indeed considering such an extension of semantic links does not longer require a simple matrix but, obviously, a more complex model. However the latter remark seems to be a very interesting idea to follow in order to improve the computational complexity of the composition process.

Another weakness of the latter approach may concern the potential involvement of the end-user during the computation of robust Web service compositions hence a supervision of the robust composition process.

Since the introduced approach is depending on the semantic descriptions of the functional input and output parameters of Web services, such an approach fails in case the latter description is not known.

Towards this issue, the approach presented in Section 4.2 extends expressivity of Web services, formally by considering preconditions and effects of Web services. This ensures to consider more composability criteria for Web service composition.

Some Concluding Remarks

In this section we suggest to study AI planning based Web service composition. More specifically, we first presented the Ra_4C process, which is able to compute correct, consistent and complete services composition by means of an SLM and an automated regression-based approach (**require-**

ment $R^{Composition}_{Automation}$). Instead a regression-based approach, other problem-solving techniques - called heuristic reasoning - may be applied [75] such as a progression-based approach [116].

The introduced approach focuses on i) input and output parameters as Web service description (a part of **requirement** $R^{Service}_{Expressivity}$) and ii) semantic links (**requirement** $R^{Semantic}_{Composability}$) as composability criteria.

Besides an automated method for Web service composition we overcome the problem of robustness in service composition by means of the Robust Ra_4C algorithm. Contrary to Ra_4C which does not consider non robust semantic links as a special case of semantic links, we have considered a method to obtain more robust compositions of Web services on the fly i.e., throughout the computation of potential (and non robust) compositions. This approach can be easily adapted to many other approaches of functional level composition.

The result of the Ra_4C process and its robust form (Robust Ra_4C) is a set of service compositions we can model as a partial order of services interleaved by semantic links. The application of the latter approaches on the flexible SLM model ensures to obtain expressive composition of Web services (requirement $R^{Composition}_{Expressivity}$) i.e. non deterministic choice of Web services, sequential and concurrent compositions. Indeed an SLM contains all required information about complete plans.

Given an SLM of a domain, most of its entries are not all necessarily required to perform "only one" composition goal. However, given different goals, its SLM can be re-used together with an AI planning approach without new pre-computation of semantic links to achieve these goals. In this direction only the *goal* columns of the SLM are required to be updated. This obviously improves the performance of the composition approach in case more than one composition goal requires to be achieved in a domain. In case two composition goals are given on the fly with two different domains, the composition approach is the same but the SLM requires to be updated (**Requirement** $R_{Flexibility}$ - very simple in most of cases, see Section 3.2) with the relevant Web services.

4.2 Causal Law based Web Service Composition

In this section we focus on a automated (**requirement** $R^{Composition}_{Automation}$) semantic link (**requirement** $R^{Semantic}_{Composability}$) and causal law (**requirement** $R^{Causal}_{Composability}$) based Web service composition. To this end Web services are defined in a high level of description i.e., by means of their input, output parameters and preconditions, effects (**requirement** $R^{Service}_{Expressivity}$). The **requirement** $R^{Composition}_{Expressivity}$ is supported by considering sequential and conditional compositions. In the same direction as approach presented in the previous section, the hard assumption related to the persistence of information (see Section 2.1.3 and models presented by [139] and [191]) is disregarded.

The main differences with the approach suggested in Section 4.1, are about:

- **Requirement** $R^{Service}_{Expressivity}$. This section focuses on compositions with more expressive description of Web services[2]. Indeed we cannot limit longer Web services to be described by only input an output parameters (Section 4.1). Therefore, by considering preconditions

[2]Such a level of description restricts and then specializes the application domain of Web services due to the addition of constraints on the functional parameters e.g., preconditions that need to be satisfied

and effects we ensure the executability of a service, and can reason about the world state after its execution. In such a context compositions comprise not only information-providing services but also world-altering Web services (i.e., services which require preconditions and provide side effects), and of course a mix of both[3].

- **Requirement** $R_{Composability}$. Composability criteria related to their semantic links (requirement $R^{Semantic}_{Composability}$) and causal laws (requirement $R^{Causal}_{Composability}$) are both concerned in the following section whereas only semantic links was concerned in Section 4.1 (See assumption in Section 4.1.1). Considering both composability criteria in Web service composition further restricts the set of potential compositions of Web services. Indeed the latter criteria are appropriate to prune large domains wherein many compositions are possible.
- **Requirement** $R^{Composition}_{Expressivity}$. In this Section the computed compositions can be modelled by sequential and conditional composition. Here the non determinism holds on **conditional output parameters** provided by Web services rather than on the choice of *Web services* that can be composed. Contrary to the approach of Section 4.1, concurrency is not addressed in this second approach.

According to these new requirements, another formalism and methodology are required to capture and exploit i) the expressive description of Web services and ii) the composability criteria related to both semantic links and causal laws to be compliant with **Requirements** $R_{Composability}$ **and** $R^{Service}_{Expressivity}$. To this end an augmented and adapted version of the logic programming language sGolog [103] (i.e., an offline interpreter that supports **Requirement** $R^{Composition}_{Expressivity}$) i.e., s_{sl}Golog is presented as a natural formalism not only for reasoning about conditional parameters of Web services, the semantic links and causal laws, but also for automatically composing services. From this, we compute Web service composition as legal execution of services conditioned on the possible output parameters.

In more details, our extension aims at

- supporting n-ary information providing services in Web service composition;
- automating branching conditional output to input parameters by means of their valid (or/and robust) semantic links together with their causal laws.

The rest of this section is organized as follows. First of all Section 4.2.1 draws the context of this Section and describes the level of *non determinism* we consider in this approach. Section 4.2.2 refines the definitions of Web services described in Section 3.2. Section 4.2.3 reviews briefly the situation calculus together with its causal laws and Golog. Section 4.2.4 extends Golog with semantic links axioms to cope with valid services. In section 4.2.5 we introduce s_{sl}Golog (= Golog + sensing + semantic link) and extend the standard backward chaining approach to compute Web service composition. Finally Section 4.2.6 highlights some limitations and gives some concluding remarks.

4.2.1 Context and Conditional Composition

Here, we remind the context of this Chapter, which is different of the context of Section 4.1. Then we present the subtle meaning differences between the *non determinism* on *Web services* (presented in Section 3.3) and the *non determinism* on *conditional output parameters* provided by Web services (presented in this Section).

[3]See Section 1.3.1 for more information about such Web services.

Our Context: Incomplete Information

Towards the issue of this Section, we conceive Web service composition as a planning and execution task, in the same way as first introduced by [139]. In such an approach the actions[4] can be complex and non deterministic [138].

As a planning task, Web service composition is then a particular context in that it is planning with very incomplete information. In this direction the context of this Section assumes domains with *incomplete information*. Contrary to Section 4.1 we draw an important difference between information-providing services and world-altering services. Here, information-providing services (i.e., information sensors) are required to provide further information and then to deal with the domains of *incomplete information*.

Moreover world-altering services are required to state about the world. In this direction we assume that Web service composition requires not only information-providing services but also world-altering services, and mix of both. Therefore, besides input and output parameters of services which act as knowledge preconditions and effects, Web services can be actions described by means of non-knowledge preconditions and (side-)effects acting in the world in a planning context. In this direction Web services can have precondition on their input parameters, and effects of their output parameters. The **requirement** $R^{Service}_{Expressivity}$ is then supported.

Non Determinism in Web Service Composition

a) Non Determinism on output parameters of Web services (addressed in this Section)

In this Section the non determinism of actions concerns the semantic description of Web services' output parameters. Given a service that provides an output parameter (defined through its semantic description), the latter parameter can be instantiated by any instance of any subsumed concept.

Example 27. ***(Non Determinism on Parameters of Web Services)***
Let `AdslEligibility` *be the Web service* S_a *described in Example 5 and Figure 1.7(a). Such a service returns a* `NetworkConnection` *of a given zone. According to the domain ontology in Figure 1.4 it is obvious that* S_a *can return an instance of* `SlowNetworkConnection` *(Figure 4.4(b)) or* `FastNetworkConnection` *(Figure 4.4(c)) as well. Indeed the latter concepts are related to* `NetworkConnection` *by the subsumption relation i.e.,*

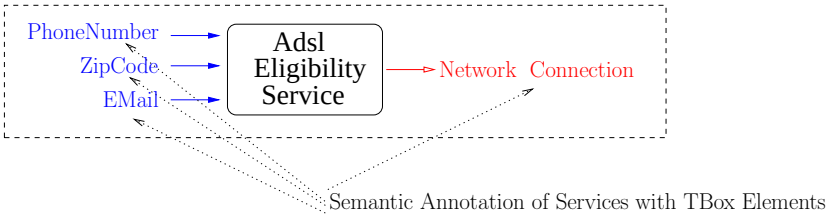
$$\texttt{SlowNetworkConnection} \sqsubseteq \texttt{NetworkConnection}$$
$$\texttt{FastNetworkConnection} \sqsubseteq \texttt{NetworkConnection}$$

Given these relationships, the service S_a *is said non determinist.*

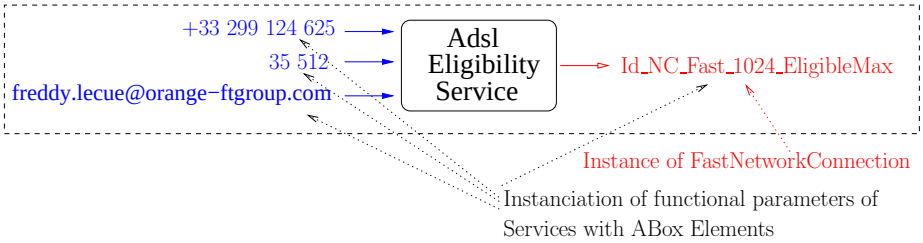
Given an output parameter and depending on the application context, any Web service can return different information for this parameter, which may cause different compositions. Here we consider that these compositions so called *conditional* compositions need to be studied in order to overcome main issues in Web service composition. In other words any relevant composer needs to adapt its methodology to consider such non determinist Web services in the composition result. To this end all potential output parameter of services will be *semantically* "branched" (or linked) to other services. Towards this issue conditional compositions aims at supporting non determinism on parameters of of Web services.

Note that this level of non determinism is not addressed in Section 4.1.

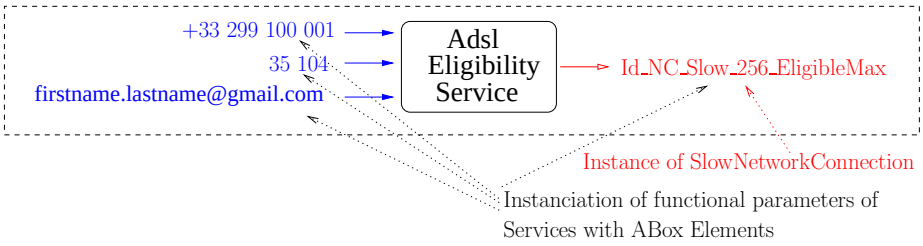
[4] Throughout this section, "action" and "service" are used synonymously.



(a) Semantic Description of a Web Service.



(b) Instanciation of a Web Service at Run Time (1).



(c) Instanciation of a Web Service at Run Time (2).

Figure 4.4: Illustration of Non Determinism on Web Services (Section 4.1).

b) Non Determinism on Web services supported by Section 4.1

In Section 4.1, non determinism is not addressed on conditional parameters but on choice of Web services. In other words, given an input parameter of a Web service, *several* Web services (that compute the same category of output parameters) could be (*semantically*) linked to the latter input parameters according to *several* semantic links. This means that a Web services could be linked to many other services, but only one have to be selected. The non determinism on Web service acts on this feature.

Example 28. ***(Non Determinism of Web Services)***
Suppose S_a, S_a^-, S_a^+ and S_c be four Web services described in Example 4 and Table 3.3. According to Section 4.1, S_c can be composed with

- *S_a, then $S_c \circ S_a$;*
- *S_a^-, then $S_c \circ S_a^-$;*

- *or* S_a^+, *then* $S_c \circ S_a^+$.

  This defines the non determinist choice on Web service S_a, S_a^-, S_a^+ *to compose with* S_c.

4.2.2 Revisited Definitions and Examples of Web Services

Since we consider information-providing together with world-altering services to perform composition we suggest to revisit Web services of Table 3.3. In this direction we follow Examples 6 and 7 illustrated in Section 1.3.1.

In addition to functional input and output parameters and semantic links (see Section 4.1) with some Web services, Web services are further described by means of some i) preconditions, effects, ii) causal laws i.e., causality relationships between effects and preconditions of service, and complex relationships between services. In this direction the revisited Web services, illustrated in Tables 4.2 and 4.3, can have causal laws i.e.,

i) causality relationships between effects and preconditions since these services may have

   - precondition axioms (e.g., the input parameter of `VoiceOverIP`* S_b^* in Table 4.2 have to be a valid network connection);
   - effect axioms (e.g., the output parameter of `VoiceOverIP`* S_b^* in Table 4.2 have to be a valid VoIPId).

ii) explicit and complex relationships with parameters of other services (e.g., a phone number needs to be attached to the output parameter of `VoiceOverIP`* S_b^* in Table 4.2);

According to Table 4.3, the `Delivery` service S_f is the only service with conditional effects.

Remark 4. ***(Complex Relationships between Web services)***
By considering causal laws related to complex and explicit relationships between Web services, some input parameters are then implicit. According to Tables 4.2 and 4.3 the `VoiceOverIP`* *and* `TvOverIP`* *services link their input parameters to a parameter which is Phone Number. This parameter is not defined as an input parameter in* `VoiceOverIP`* *and* `TvOverIP`* *but required as an implicit parameter through a causal law.*

The formal definition of axioms related to *preconditions* and *causal laws* will be detailed in the next section.

Definition 21. ***(Output Parameters of Services)***
The n^{th} *parameter of Web service* s_x *is noted as*

$$output(s_x(y_1, ..., y_n), n) \tag{4.2}$$

wherein $(y_1, ..., y_n)$[5] *refers to input parameters of Web services* s_x.

This formalism is suggested in this section since it is more appropriate to model output parameters of Web services based AI planning actions.

In Tables 4.2 and 4.3 as well as in the rest of this section, output parameters of any services s_x will be refined by means of Definition 21.

In addition we formalize the *knowledge of a referent* [177] by Definition 22.

[5] Throughout this section, $(y_1, ..., y_n)$ and $\vec{y}$ are used synonymously.

Definition 22. ***(Knowledge of a Referent)***
When the value of a term is known, we use the notation $Kref(v)$. *This informs about the knowledge of a value* v_i *for the term* v.

The latter definition is used to model, for instance, information which is required by Web services, before their executions.

	Services s_x	`AdslEligibility`* S_a^*	`VoiceOverIP`* S_b^*	`TvOverIP`* S_c^*	`Billing` S_d
Functional Parameters	$Input(s_x)$	$\{ph : PhoneNumber, zc : ZipCode, e : Email\}$	$\{snc : SlowNetworkConnection\}$	$\{fnc : FastNetworkConnection\}$	$\{d : Decoder\}$
	$output(s_x(\vec{y}), n)$ i.e., $Output(s_x)$	$output(S_a^*(ph, zc, e), 1)$ i.e., $\{nc : NC\}$	$output(S_b^*(snc), 1)$ i.e., $\{vid : VoIPId\}$	$output(S_c^*(fnv), 1)$ i.e., $\{vd : VideoDecoder\}$	$output(S_d(d), 1)$ i.e., $\{i : Invoice\}$
Functional	$Precondition(s_x)$	$FrenchPhone(ph)$, $FrenchZip(zc)$, $validMail(e)$	$validNetworkConnection(snc)$, $supportedConnectionType(snc)$, $Kref(NetworkConnection(x))$	$validNetworkConnection(fnc)$, $supportedConnectionType(fnc)$, $Kref(NetworkConnection(x))$	$validDecoder(d)$
	$Effect(x)$	$validNC(nc)$	$validIPId(vid)$	$validVD(vd)$	$validIn(i)$
Laws	Causality Relationships between Effects of Actions and $Precondition(s_x)$	Inferred at composition time	Inferred at composition time	Inferred at composition time	Inferred at composition time
Causal	Complex Relationships between Actions	n.a	if snc related to a Phone Number ph then vid is related to ph	if fnc related to a Phone Number ph then vd is related to ph	if d related to a Phone Number ph then i is related to ph

Table 4.2: Revisited Description of Web services defined in Table 3.3

In Tables 4.2 and 4.3 we focused on specific inputs, outputs, preconditions, effects and causal laws of a reduced set of Web services. Input, output parameters together with preconditions and effects of `AdslEligibility`* S_a^* are illustrated in Figure 7.2 of Chapter 7.

4.2.3 Background: Situation Calculus and Golog

The situation calculus and Golog are used as logic formalisms to describe Web service composition and compute its solutions. The expressive power and formal semantics of Golog provide the theoretical foundations i) for reasoning on causal laws, and ii) for the encoding of semantic links.

Situation Calculus

Since the suggested approach is based on the *Situation Calculus* formalism, we preferred introducing the latter formalism in this Section rather than in part I (Related Work part).

There are a number of ways of making the planning based composition precise, but perhaps the most appealing is to formulate a specification in terms of a general theory of action. One candidate language for formulating such a theory is the situation calculus [177]. We will not go over the language here, but only review important components of the language. Briefly, the situation calculus is a first-order language with some limited second-order features specifically

	Services s_x	`Availability` S_e	`Delivery` S_f	S_g	S_h
Parameters	$Input(s_x)$	$\{ipa : IPAddress\}$	$\{i : Invoice\}$	$\emptyset$	$\emptyset$
	$output(s_x(\vec{y}), n)$ i.e., $Output(s_x)$	$output(S_e(ipa), 0)$ i.e., $\emptyset$	$output(S_f(i), 1)$ i.e., $\{d\_id : DeliveryID\}$	$output(S_g(), 0)$ i.e., $\emptyset$	$output(S_h(), 0)$ i.e., $\emptyset$
Functional	$Precondition(s_x)$	$validIPA(ipa)$	$validIn(i)$	$success(S_f)$	$failure(S_f)$
	$Effect(s_x)$	$remove\text{-}AvailableIPA(ipa)$	$remove\text{-}AvailableDecoder(d)$, $success(S_f) \vee failure(S_f)$	n.a	n.a
Laws	Causality Relationships between Effects of Actions and $Precondition(s_x)$	Inferred at composition time	Inferred at composition time	Inferred at composition time	Inferred at composition time
Causal	Complex Relationships between Actions	n.a	if i related to a Decoder d then d_id is is related to d	n.a	n.a

Table 4.3: Illustration of New Services with Extended Descriptions.

designed for representing dynamically changing worlds in which all changes are the result of named *actions*. In the following *action* will refer to *service* (and vice versa) without loss of generality. The language has a special constant S_0 used to denote the initial situation where no actions have occurred yet. There is a distinguished binary function symbol do where $do(a, s)$ denotes the successor situation to s resulting from performing action a. The state of the world is expressed in terms of functional and relational *fluents* relativized to a particular situation s, e.g., $F(\vec{y}, s)$. A special predicate $Poss(a, s)$ is used to state that action a is executable in situation s. Actions may have sensing results, and the function $sr(a, s)$ denotes the sensing result of action a when executed in situation s [183].

Within this language, we can formulate domain theories which describe how the world changes as the result of the available actions. One possibility is a variant of the basic action theory [176] of the form $\mathcal{D}$:

$$\mathcal{D} = \Sigma \cup \mathcal{D}_{una} \cup \mathcal{D}_{ap} \cup \mathcal{D}_{S_0} \cup \mathcal{D}_{ss} \cup \mathcal{D}_{sr} \tag{4.3}$$

- Σ is the set of *domain-independent, foundational axioms* for situations. In a nutshell this set consists of four axioms i.e., i) a unique names axiom for situations, ii) the second axiom is second order induction on situations (i.e., the domain closure axiom for situations), iii) the third and forth axioms are related to an ordering relation on situations. They provide the basic properties of situations in any domain specific axiomatization of particular fluents and actions.

- $\mathcal{D}_{una}$ is the set of *unique name axioms* for primitive actions, stating that the actions of the domain are pair wise unequal. For distinct action function symbols a and b,

$$a(x_1, ..., x_n) \neq b(y_1, ..., y_n). \tag{4.4}$$

  Identical action terms have identical arguments:

$$x_1 = y_1 \wedge ... \wedge x_n = y_n \leftarrow a(x_1, ..., x_n) = a(y_1, ..., y_n) \tag{4.5}$$

- $\mathcal{D}_{ap}$ is the set of *action precondition axioms*, one for each primitive action a, characterizing $Poss(a, s)$ i.e., the preconditions of a. Formally, an action precondition axiom is a sentence of the form:

$$Poss(a(x_1, ..., x_n), s) \equiv \prod_a (x_1, ..., x_n, s), \tag{4.6}$$

where a is an n-ary function symbol, and $\prod_a(x_1, ..., x_n, s)$ is a formula that is uniform in s and whose free variables are among $x_1, ..., x_n, s$. The uniformity requirement on $\prod_a$ ensures that the preconditions for the executability of the action $a(x_1, ..., x_n)$ are determined only by the current situation s, not by any other situation.

Example 29. ***(Illustration of $\mathcal{D}_{ap}$)***
Let `VoiceOverIP`* S_b^* *be the service illustrated in Table 4.2. This service is defined in the same way as* `VoiceOverIP` *in Example 4 and Table 3.3 but does not require* `PhoneNumber` *as input parameter but only a* `SlowNetworkConnection`*. From this, in the Telecommunication world, we might typically have preconditions on this action as follows:*

$$\begin{aligned} Poss(\texttt{VoiceOverIP}^*(x), s) \equiv \\ validNetworkConnection(x, s) \wedge \\ supportConnectionType(x, s) \end{aligned} \tag{4.7}$$

In other words `VoiceOverIP`* *is possible in situation s in case its parameter x is a valid network connection, and x is supported by the action* `VoiceOverIP`**.*

- $\mathcal{D}_{S_0}$ is a set of axioms describing the *initial situation* S_0 and axioms not mentioning situations at all. $\mathcal{D}_{S_0}$ is then a set of first order sentences that are uniform in S_0. Thus, no sentence of $\mathcal{D}_{S_0}$ quantifies over situations, or mentions $Poss$, the ordering of situations or the function symbol do, so that S_0 is the only term of sort *situation* mentioned by these sentences. $\mathcal{D}_{S_0}$ will function as the initial theory of the world (i.e., the one we start of with, before any actions have been "executed"). Often, we shall call $\mathcal{D}_{S_0}$ the *initial database*.

Example 30. ***(Illustration of $\mathcal{D}_{S_0}$)***
Given a Telecommunication world, some "timeless" *facts like* $isZipCode(35512)$, $isEmailAddress(freddy.lecue@orange\text{-}ftgroup.com)$ *are known in situation S_0. As we will see in the following, such facts are inferred from instance checking through DL reasoning. Moreover some facts are true in situation S_0 e.g.,* $FrenchPhone(+33299124625, S_0)$, $FrenchZip(35512, S_0)$, $validMail(freddy.lecue@orange\text{-}ftgroup.com, S_0)$.

- $\mathcal{D}_{ss}$ is the set of *successor state axioms*, one for each fluent F in the domain, of the form

$$F(\vec{y}, do(a, s)) = v \equiv \Phi_F(\vec{y}, a, v, s) \tag{4.8}$$

where $\Phi_F(\vec{y}, a, v, s)$ is a formula uniform in s, all of whose free variables are among $a, s, y_1, ..., y_n$. The latter axioms state the conditions under which the fluent has a specific value at situation $do(a, s)$ as a function of situation s. These axioms take the place of the so-called causal laws, which are causality relationships between effects and preconditions, and complex relationships between actions and constraints between actions preconditions and effects. Moreover they also provide a solution to the frame problem [176].

Example 31. ***(Illustration of $\mathcal{D}_{ss}$)***
An example of such an axiom is as follows:

$$\begin{aligned} phoneNumberOf(output(\texttt{VoiceOverIP}^*(x),1),ph\_nb,do(\texttt{VoiceOverIP}^*(x),s)) \leftarrow \\ Poss(\texttt{VoiceOverIP}^*(x),s) \wedge \\ phoneNumberOf(x,ph\_nb,s) \vee \\ phoneNumberOf(output(\texttt{VoiceOverIP}^*(x),1),ph\_nb,s) \end{aligned} \quad (4.9)$$

wherein x, s, ph_nb are some variables.

Axiom (4.9) says that ph_nb will be the phone number of the output parameter provided by `VoiceOverIP`* *in the successor situation $do($*`VoiceOverIP`$^*(x), s)$ if* `VoiceOverIP`* *was a possible action in situation s and ph_nb was the phone number of x (i.e., the input parameter of* `VoiceOverIP`**) in situation s, or ph_nb was already the phone number of the output parameter provided by* `VoiceOverIP`**. Such an axiom can be illustrated by means of Figure 4.5.*

- $\mathcal{D}_{sr}$ is a set of *sensing-result axioms* (SRAs), one for each sensing action symbol a, of the form $sr(a(\vec{x}), s) = r \equiv \Theta_a(\vec{x}, r, s)$. SRAs relate sensing output parameters with fluents.

Example 32. ***(Illustration of $\mathcal{D}_{sr}$)***
Let `AdslEligibility` *S_a be exactly the service illustrated in Example 4 and Table 3.3. The referent of the output parameter $NetworkConnection$ of this service is known by means of the latter axiom.* `AdslEligibility` *is a binary sensing action such that $sr($*`AdslEligibility`$, s)$ *be $NetworkConnection(x, s)$.*

The set of *sensing-result axioms* are, in many cases, attached to appropriate $Kref$-terms [177]in the form of $Kref(v, s)$. $Kref(v, s)$ is a situation-based adaptation of Definition 22, which informs about the knowledge of a value v_i for the term v in situation s.

Example 33. ***(Knowledge Referent)***
Suppose the `VoiceOverIP`* *action defined in Table 4.2. Such a service requires a* `SlowNetwork-Connection` *as input parameter. Since* `AdslEligibility` *is a binary sensing action such that $sr(AdslEligibility, s)$ be $NetworkConnection(x, s)$ (see Example 32), its referent can be bound to the input parameter* `SlowNetworkConnection` *of action* `VoiceOverIP`**.*

Therefore these axioms further constrain, in particular, the set of *action precondition axioms* $\mathcal{D}_{ap}$.

Example 34. ***(Illustration of Further constraints on $\mathcal{D}_{ap}$)***
Suppose example 29. The revisited preconditions of the `VoiceOverIP`* *S_b^* action are as follows:*

$$\begin{aligned} Poss(\texttt{VoiceOverIP}^*(x),s) \equiv \\ validNetworkConnection(x,s) \wedge \\ supportConnectionType(x,s) \wedge \\ \boldsymbol{KRef}(NetworkConnection(x),s) \end{aligned} \quad (4.10)$$

In other words `VoiceOverIP`* *is possible in situation s in case its parameter x is a valid network connection and x is supported by the action* `VoiceOverIP`**. Moreover an axiom related to the preconditions of action* `VoiceOverIP`* *is defined by $Kref(NetworkConnection(x), s)$. According to the latter constraint,* knowing the referent of $NetworkConnection(x)$ in situation s is not a truth about the physical world, but about the knowledge base of the composer executing the action `VoiceOverIP`*.

The referent of $NetworkConnection(x)$ can be known in situation s in case the action `AdslEligibility` *was possible and has been executed in a previous situation.*

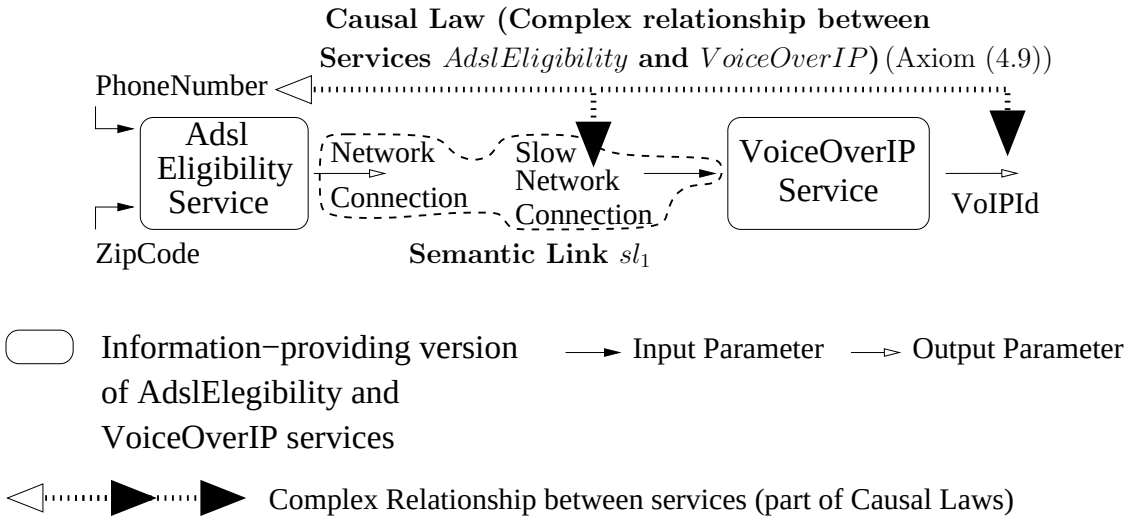


Figure 4.5: A Trivial Composition of two Actions with a Complex Relationship.

Golog

Golog [121] is a high-level logic programming language for the specification and execution of complex actions in dynamic domains. It builds on top of the situation calculus by providing extra logical constructs for assembling primitive situation calculus actions, into complex actions δ. Constructs of Golog are defined in Figure 4.6.

a - primitive actions
$\delta_1; \delta_2$ - sequences
$\phi?$ - tests
$\delta_1 \vert \delta_2$ - nondeterministic choice of actions
$(\pi x)\delta(x)$ - nondeterministic choice of arguments
δ^* - nondeterministic iteration
if ϕ **then** δ_1 **else** δ_2 **endif** - conditionals
while ϕ **do** δ **endwhile** - while loops
proc $\delta(x)$ **endproc** - procedure

Figure 4.6: Some Golog Constructs.

The constructs illustrated in Figure 4.6 can be used to write programs in the language of the domain theory. More specifically they can be used to specify Web service compositions with their semantic links and causal laws.

Example 35. *(Web service composition with Golog)*

Let the following example[6] *be a composite Web service.*

$$getNetworkConnection(\vec{x});$$
$$\textbf{if } slow \textbf{ then } voiceOverIP^*(\vec{y}) \textbf{ else } tvOverIP(\vec{y}) \textbf{ endif} \quad (4.11)$$

This example expresses that a service (actually a method `getNetworkConnection` *of a service) is responsible of first computing the network connection of a given geographical zone. In case the network connection is considered as* `slow` *we apply the service* `voiceOverIP`* *otherwise* `tvOverIP`**. This illustrates a conditional composition of Web services.*

Given a domain theory, $\mathcal{D}$ and a Golog program δ, its execution must find a sequence of actions $\vec{a}$ such that: $\mathcal{D} \models Do(\delta, S_0, do(\vec{a}, S_0))$ denotes that the Golog program δ, starting execution in S_0 will legally terminate in situation $do(\vec{a}, S_0)$, where $do(\vec{a}, S_0)$ abbreviates $do(a_n, do(a_{n-1}, ..., do(a_1, S_0)))$.

4.2.4 Specifying Semantic Links with Golog

Since any basic action theory of the form $\mathcal{D}$ is under specified to model semantic links between actions, this section is motivated towards an appropriate extension of basic action theories $\mathcal{D}$ to model semantic links between actions. Indeed $\mathcal{D}$ does not have an explicit account of semantic link between parameters of actions, a key concept for any service composer since such links are considered as an important issue [108] to form valid Web service compositions. For instance any regressable Web service composition (sentence in Situation Calculus) in [139, 193] is entailed by $\mathcal{D}$, without considering any axiom related to semantic links. Such a limitation makes Web service composition very hard to achieve since a composition does not require only causal laws to express the logical dependence between actions and their effects and preconditions, but also semantic links between output and input parameters of actions. By extending Golog to enable Golog programs to specify semantic links together with causal laws, we overcome the latter issue.

Methodology: Situation Calculus and DL Reasoning for Web service composition

We suggest to operate with i) DL reasoning to infer semantic links between actions (as presented in Chapter 3) and ii) situation calculus to represent Web service compositions and infer them through reasoning on their axioms in $\mathcal{D}$.

In the same way as [139], we define through the situation calculus formalism (e.g., Golog) actions $a(x_1, ..., x_n)$ in situation s. The parameters of such actions are variables representing the inputs of the actions. Indeed parameters of actions cannot be restricted to constants since i) output parameters of actions are depending on their inputs and ii) we want to distinguish output parameters coming from two (or more) different executions of an action with distinct input parameters. Each variable can be instantiated by entities; a variable enables us to denote the same entity in the state preceding and the state following the execution of the action. Moreover these variables x_i will be mapped to concepts X_i in a separated Terminological Box of a DL knowledge base. In this direction a relation $Instance()$ is introduced, and $Instance(X, x, s)$ means that x is an instance of concept X in situation s.

Example 36. *(Formalization of an Action (Service))*
Let `VoiceOverIP`* *be an action that requires only an instance of a* `SlowNetworkConnection` *(attached to a phone number - see Example 31) as input and returns an* `VoIPId` *as output (Figure*

[6] Notation: Fluents are in situation-suppressed form. Variables are universally quantified unless otherwise noted.

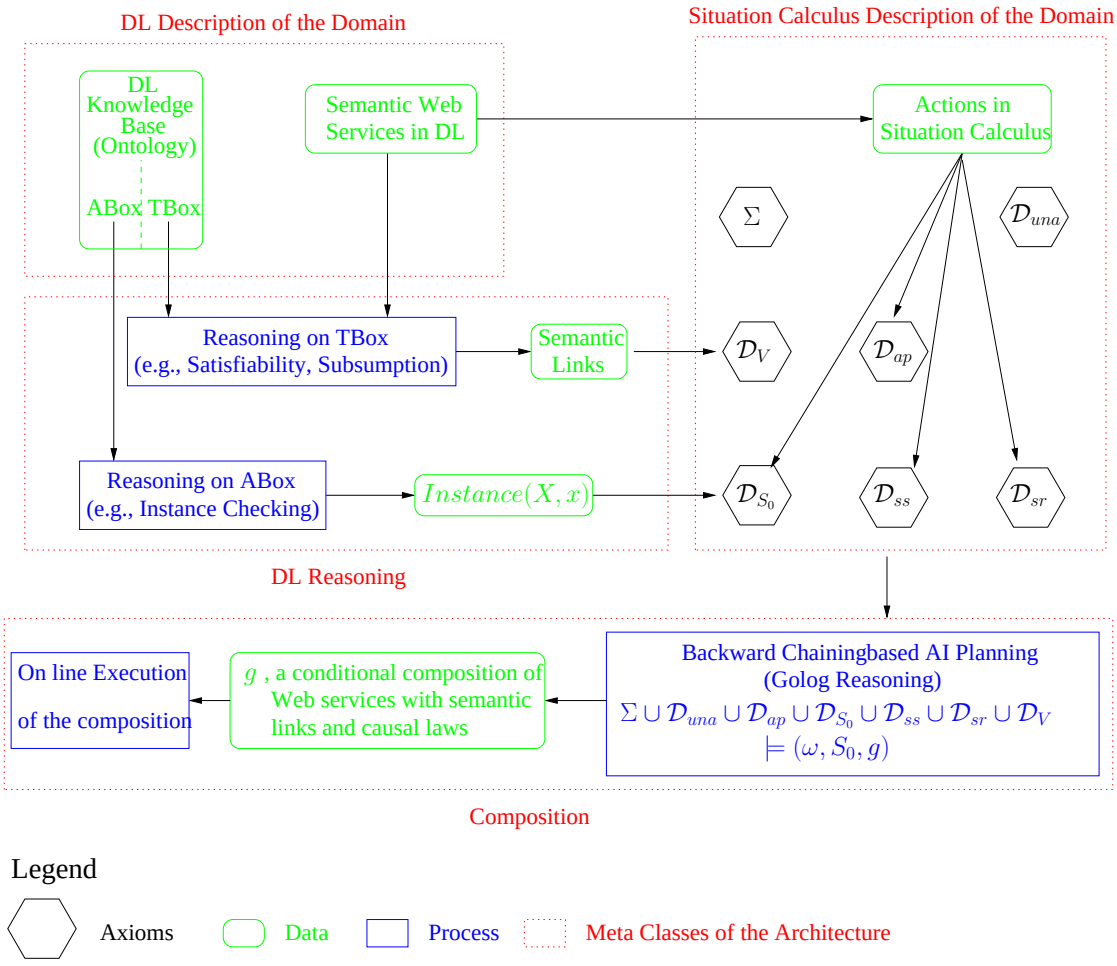


Figure 4.7: Architecture/Methodology of the Overall Approach.

4.5). By considering a DL knowledge base $\mathcal{T}$, concepts such as `VoIPId`, `SlowNetworkConnection` *and* `NetworkConnection` *(i.e., subsumer of* `SlowNetworkConnection`*) are defined in its Terminological Box (Figure 1.4); the Assertional Box contains instances of these concepts.*

Since the semantic descriptions of action parameters are outside of the situation calculus formalization, DL reasoning (e.g., through subsumption) with the latter descriptions is required to be performed. This is achieved in the approach we follow along this section illustrated in Figure 4.7.

Then, situation calculus axioms related to these DL reasoning results are generated and integrated in the action theories $\mathcal{D}$ to achieve Web service composition. The axiom $\mathcal{D}_V$ refers to semantic links we infer in a first step of DL reasoning on the Terminological Box; the axiom $\mathcal{D}_{S_0}$ is in part inferred from DL reasoning on the Assertional Box and initial conditions in the composition problem.

Valid Action and its Semantic Links

We introduce a new distinguished fluent in the situation calculus called $Valid(a, s)$ e.g., each input parameter of action a is related to an output parameter of another action in situation s. We contrast this with $Poss(a, s)$ i.e., action a is physically possible in situation s. We further restrict the cases in which an action is executable by requiring not only that an action a is $Poss(a, s)$ but further that it is $Valid(a, s)$ i.e.,

$$executable(a, s) \doteq Poss(a, s) \wedge Valid(a, s) \tag{4.12}$$

This further constrains the search space for actions when realizing a Golog program. The set of $Valid$ fluents, one for each action, is referred to as $\mathcal{D}_V$. $Valid(a, s) \equiv true$ unless otherwise noted. The constraints of action a and its input parameters $x_{i,1\leq i\leq n}$ are expressed in the situation calculus as *necessary conditions for an action a to be valid*, $\mathcal{D}_V$:

$$Valid(a(x_1, ..., x_n), s) \leftarrow \bigwedge_{i=1}^{n} \Omega_a^{SLC(x_i)}, \tag{4.13}$$

where the i^{th} *semantic link constraints* of a are defined by $\Omega_a^{SLC(x_i)} = SemanticLink(Y_i, X_i, s) \wedge Instance(Y_i, x_i, s)$. $SemanticLink$ is a distinguished predicate[7], relating two DL descriptions Y_i and X_i in situation s. More precisely, Y_i is the set of DL descriptions satisfied by the referent bound to variable x_i in situation s. $SemanticLink(Y_i, X_i, s)$ aims at computing the existence or absence of a semantic match between Y_i and X_i. Since we consider valid Web service composition (Definition 12 in Section 3.1) the latter semantic match is restricted to its *valid* match types (i.e., match types that define valid semantic links, Definition 11) i.e.,

- *Exact* (i.e., $\mathcal{T} \models Y_i \equiv X_i$);
- *PlugIn* (i.e., $\mathcal{T} \models Y_i \sqsubseteq X_i$);
- *Subsume* (i.e., $\mathcal{T} \models X_i \sqsubseteq Y_i$);
- *Intersection* (i.e., $\mathcal{T} \not\models Y_i \sqcap X_i \sqsubseteq \bot$).

Axioms related to the validity of semantic links are then required to infer valid Web service compositions.

To this end all valid semantic links between actions are computed either in a pre-processing step or when needed (due to its complexity in large problem) by means of an external Description Logics reasoner such as Fact++ [92]. Formally a set of semantic links between two DL descriptions are inferred from the valid match type m_t as follows:

$$SemanticLink(Y, X, s) \leftarrow Valid_{matchType}(m_t, s), \forall s \tag{4.14}$$

where $Valid_{matchType}(m_t)$ is satisfiable in case m_t is a valid match type, hence a valid semantic link between Y and X. By means of (4.14) it is straightforward to compute all valid semantic links in any situation s. Having computed $\mathcal{D}_V$, we include it in $\mathcal{D}$. Henceforth, all reference to $\mathcal{D}$ includes $\mathcal{D}_V$.

Example 37. *(Valid Action (Service) in Golog)*
Here we illustrate valid actions with the simple composition in Figure 4.5. According to a first

[7]Here, the Definition 10 of *semantic link* has been adapted by the *SemanticLink* predicate for encoding in the Situation Calculus formalism.

step of DL reasoning and from axiom (4.15), semantic links valued by a Subsume match type are inferred. In other words any pair of output and input parameters of services that match with a Subsume match type are computed by means of axiom (4.15). For instance the pair (`NetworkConnection`, `SlowNetworkConnection`) *is retrieved.*

$$SemanticLink(\texttt{NetworkConnection},\texttt{SlowNetworkConnection}, s) \leftarrow Valid_{matchType}(Subsume, s) \quad (4.15)$$

From this, we can infer the validity of action `VoiceOverIP`* *by means of axiom 4.13. Therefore we obtain axiom (4.16). More specifically the input parameter x of* `VoiceOverIP`* *have to be an instance of a concept semantically close to (here Subsume) the input parameters that* `VoiceOverIP`* *requires. The semantic closeness is valued by the valid match types we approve, here Subsume.*

$$\begin{aligned} Valid(\texttt{VoiceOverIP}^*\ (x), s) \leftarrow\ & \\ & SemanticLink(Y, \texttt{SlowNetworkConnection}, s) \wedge \\ & Instance(Y, x, s) \end{aligned} \quad (4.16)$$

In addition to the previous axioms, axioms related to causal laws (i.e. successor state axiom) (4.17) and (4.18) are also required. Indeed a composer executing `VoiceOverIP`* *under some conditions (see body of (4.17) and (4.18)) causes its output parameter i) to be related to a phone number ph_nb (*through Golog reasoning *on causal laws) and ii) to be an instance of* `VoIPId` *(*through DL reasoning*) in the successor state.*

$$\begin{aligned} phoneNumberOf(output(\texttt{VoiceOverIP}^*\ (x), 1), ph\_nb, do(\texttt{VoiceOverIP}^*\ (x), s)) \leftarrow\ & \\ & Poss(\texttt{VoiceOverIP}^*\ (x), s) \wedge \\ & Valid(\texttt{VoiceOverIP}^*\ (x), s) \wedge \\ & phoneNumberOf(x, ph\_nb, s) \vee \\ & phoneNumberOf(output(\texttt{VoiceOverIP}^*\ (x), 1), ph\_nb, s) \end{aligned} \quad (4.17)$$

wherein x, s, ph_nb are some variables.

Here axiom (4.17) is an extension of axiom (4.9). By extending (4.9) we ensure that action `VoiceOverIP`* *is valid.*

$$\begin{aligned} Instance(\texttt{VoIPId},\ output(\texttt{VoiceOverIP}^*\ (x), 1)\ , do(\texttt{VoiceOverIP}^*\ (x), s)) \leftarrow\ & \\ & Poss(\texttt{VoiceOverIP}^*\ (x), s) \wedge \\ & Valid(\texttt{VoiceOverIP}^*\ (x), s) \wedge \\ & phoneNumberOf(x, ph\_nb, s) \vee \\ & Instance(\texttt{VoIPId},\ output(\texttt{VoiceOverIP}^*\ (x), 1)\ , s) \end{aligned} \quad (4.18)$$

wherein x, s, ph_nb are some variables.

Let nc, ph be respectively two instances (i.e., elements in the ABox) of `NetworkConnection` *and* `PhoneNumber` *in situation s i.e., $Instance(\texttt{NetworkConnection}, nc, s)$ and $Instance(\texttt{PhoneNumber}, ph, s)$ hold in any situation s. Semantic links valued by a Subsume match type are identified by (4.15) and valid actions by (4.16). Therefore $Valid(\texttt{VoiceOverIP}^*\ (nc), s)$ is inferred.*

Finally, some facts related to output(`VoiceOverIP`* $(nc), 1)$ *are inferred from* (4.17) *and* (4.18). *Note that the* `NetworkConnection` *nc is bound to the* `SlowNetworkConnection` *parameter of* `VoiceOverIP`*, *as there exists a semantic link between them.*

Remark 5. ***(Valid Action: Extension to Other Actions)***
As previously said Example 37 focuses on a simple composition of two services and on a validity of a service `VoiceOverIP`*. *However considering a composition of more Web services requires that any involved service satisfies the definition of valid action. In this direction a related work needs to be achieved on* S_a^*, S_b^*, S_c^*, S_d, S_e, S_f, S_g *and* S_h *in case they are all involved in the final composition.*

In the Telecommunication scenarios considered in this work, the DL descriptions of parameters do not use the relations that the Golog reasoning (as *phoneNumberOf* in (4.18) and (4.17)) will infer. This ensures that subsumption reasoning has not to use the facts that are true in a given state but use only the axioms declared in the Tbox. Furthermore this ensures that DL reasoning is independent of the situation calculus states and can thus be performed by an independent external DL reasoner.

Besides extending expressivity of Web services by considering their preconditions, effects and causal laws, the semantic link definition, introduced in Chapter 3, has been refined by means of axiom (4.13) especially to support trivial semantic links in S_0. Therefore input parameter x_i of action a can be bound not only to any output parameter of any action but also to instances in initial situation S_0.

Example 38. ***(Valid Action in Situation*** S_0***)***
According to Example 37, the `VoiceOverIP`* *service is valid since its input parameter is semantically related to an output parameter of another service* `AdslEligibility`. *In case such a service is not available but an instance of a* `NetworkConnection` *or* `SlowNetworkConnection` *is available in situation* S_0, *the* `VoiceOverIP`* *service is still valid. Indeed the latter instance can be bound to the input parameter of* `VoiceOverIP`* .

Restricting the Search Space of Web Service Composition

In this section the semantic link between Web services together with causal laws are used to restrict the search space in Web service composition. Indeed composing Web services with such constraints is restrictive, but ensures to compute only valid and possible compositions of Web services. In case one prefers retrieving compositions of robust services (more restrictive than valid; see Definition 13), (4.13) can be adapted by simply replacing the predicate $Valid_{matchType}$ in (4.14) by $Robust_{matchType}$ (i.e., match types that define robust semantic links, Definition 13) which returns *true* in case m_t is a robust match type (i.e., Exact or PlugIn), and false otherwise (Intersection or Subsume).

The search space can be further reduced by adding some customizing constraints e.g., $Desirable$ [139] or $Preferred$ [193] actions.

Some Concluding Remarks

The definition of valid actions $\mathcal{D}_V$ may simply be added to any existing situation calculus axiomatization. In case some semantic links change (e.g., due to unavailable actions), the affected $Valid$ fluents in $\mathcal{D}_V$ can be re-generated by DL reasoning and then elaborated by a simple local rewrite.

The $Valid$ axiom is easily implemented as an augmentation of most existing Golog interpreters such as sGolog [103]. It reduces the search space for terminating situations, rather than pruning

situations after they have been found. Our approach has advantages over other approaches to determine sequences of valid actions wherein causal laws (requirement $R^{Causal}_{Composability}$) together with semantic links (requirement $R^{Semantic}_{Composability}$) hold (axiom (4.12)).

4.2.5 Adapting sGolog for Composition of Services

Since Web service composition is defined as planning with very incomplete information, conditions that hold or not at any point in the plan (composition) cannot be logically determined by the background domain theory $\mathcal{D}\backslash\mathcal{D}_{sr}$ such as the subset of theory introduced by (4.3). Towards such a limitation, incorporating information-providing actions in Web service composition and axiom $\mathcal{D}_{sr}$ seem an appropriate solution. In building a Golog interpreter that incorporates such actions, the interplay between sensing and execution of world-altering actions can be complex and a number of different approaches have been discussed. While [77] and [177] advocate the use of an online interpreter to reason with sensing actions, [103] suggests the use of an offline interpreter sGolog with conditional plans. In our approach online interpreters are disregarded for our composition task. Indeed they are incomplete since no backtracking is allowed at execution time, which is a hard limitation. Moreover the latter interpreters operate under an assumption of *reasonable persistence* [139] of information being sensed (see Section 2.1.3 and models presented by [139] and [191]).

Even if an offline interpreter is computationally expensive due to the much larger search space, it enables us to generate conditional plans without the hard constraint of *reasonable persistence* of information, if sensing actions are involved. Thus off-line interpreters seem the most appropriate to reason about sensing and hence retrieving all conditional compositions whatever information being sensed. However as focused by [103], Golog is under specified to support conditional compositions. Indeed Golog, running offline, is bound to fail since it cannot decide between different conditional branches. Towards this issue we present an offline interpreter s_{sl}Golog i.e., an extension of sGolog that i) supports n-ary sensing actions to compute conditional Web service compositions and ii) elaborates a strategy for automated branching by means of semantic links and causal laws.

Towards these issues we first present *General Conditional Action Trees* i.e., trees that model situations wherein a non determinism (or conditional) choice is available. From this definition we introduce s_{sl}Golog i.e., an extension of Golog that support *General Conditional Action Trees*. Then, we adapt *Backward Chaining* based AI planning for s_{sl}Golog in order to achieve Web service composition. From the s_{sl}Golog result (which is a conditional composition of Web services) we present how the composition is executed. Finally we expose some limitations and possible extensions.

General Conditional Action Trees

In this section, we augment the situation calculus with general conditional action trees (General CAT i.e., GCAT). Instead of having only linear action histories (i.e., situations [177]) or binary trees of actions (i.e., CAT [103] containing if-$then$-$else$ clauses), we have also n-ary trees of actions (i.e., GCAT containing *switch-case* clauses). GCATs are plans with branches, wherein their nodes are situations and the root represents the initial situation. Every edge is labelled with a primitive action, which indicates how a situation is obtained from its predecessor. In addition, whenever branching occurs, the corresponding node/situation is labelled by a formula, whose truth value at execution time determines which branch is selected.

In addition to primitive actions, a GCAT may include a special constant ϵ, denoting the empty GCAT, and two constructors $a.g$ and the multi-branching node, where a is an action and

g is a GCAT. The multi-branching node has the form $[\phi_v, (v_1, g_1), (v_2, g_2), ..., (v_m, g_m)]$, where ϕ_v is a *multi-branch-formula* determining which branch is to be executed (i.e., by retrieving v_i such that $\phi_v(v_i, s)$ be true), and $g_1, g_2, ..., g_m$ are respectively the v_1-, v_2-,... and v_m-branch.

Remark 6. ***(Simplification for the Multi-Branching Node)***
Logically, the multi-branching node is a $(m+1)$-ary function. In the following we restrict m to be 2 without loss of generality.

The introduction of the *multi-branch* construct first guarantees multiple branching in Web service compositions and second ensures that s_{sl}Golog, running offline, succeeds.

Example 39. *(Branch Formula and Execution)*
Let $g = a_1.a_2.[\phi_v, (v_3, a_3), (v_4, \epsilon)].a_5$ and let ϕ_v denoting true if v is v_4 at $do(a_2, do(a_1, s))$, and false otherwise. Then $g = a_1.a_2.[(v_4, a_5)]$ at execution time. Note that g is depending on ϕ_v.

Since the s_{sl}Golog interpreter produces GCATs we introduce the function *gdo*, which takes a GCAT g, a situation s and returns a situation which is obtained from s using the actions along a particular path in g. *gdo* follows a particular branch in the n-ary tree conditioned on the possible outcome of sensing actions (i.e., value of the corresponding *multi-branch* formula) relative to the current situation. Formally we have:

$$
\begin{aligned}
&gdo(\epsilon, s) = s.\\
&gdo(a, s) = do(a, s).\\
&gdo(a.g, s) = gdo(g, do(a, s)).\\
&gdo([\phi_v, (v_1, g_1),(v_2, g_2)], s) =\\
&\qquad\qquad \textbf{if } \phi_v(v_1, s) \textbf{ then } gdo(g_1, s)\\
&\qquad\qquad \textbf{else if } \phi_v(v_2, s) \textbf{ then } gdo(g_2, s).
\end{aligned}
$$

The term *gdo* is required in the following to extend the *do* term and then supporting multi-branching and then n-ary conditional composition of Web services.

s_{sl}Golog

Programs in s_{sl}Golog are the same as sGolog augmented by n-ary sensing actions for both formulas and terms. The main difference with sGolog is that the interpreter now produces GCATs instead of CATs. When constructing a GCAT, new branches i.e., constructs of the form $[\phi_v, (v_1, g_1), (v_2, g_2)]$ need to be introduced. However such constructs require a value to be branched (especially to value $\phi_v(v_i, s)$). That is why we suggest a new special action $multibranch\_on(v)$, whose "*effect*" is to introduce a new GCAT $[\phi_v, \epsilon, \epsilon]$ for enabling a branching to v. Thus any information-providing action can be followed (not necessarily directly) by first, a $multibranch\_on(v)$ action and second, a $switch(case(v_1, e_1), case(v_2, e_2))$ action for simulating conditional branching of term v to v_1 or v_2. The simulation of *switch-case* by if-*then-else* is straightforward.

Automated branching in a GCAT can be ensured by the axiom of valid actions. Any action in each sub-branch can use the sensed value (i.e., output parameter of an action) as an input by means of their semantic links. The causal laws need also to hold. Alternatively [103, 199, 182] required the end-user to provide the branching strategy.

Since s_{sl}Golog aims at producing GCATs ready for execution, any term v involved in a branching $multibranch\_on(v)$ has to be known. To this end, an appropriate $Kref$-term [177] (see Section 4.2.3) is attached to the definition of *multibranch_on*. As previously said $Kref(v, s)$

informs about the knowledge of a value v_i for the term v in situation s. For instance a sensing action can retrieve a value v_i for v, which is a potential output parameter of this action. Technically, the s_{sl}Golog interpreter is defined in a way very similar to Golog and sGolog. We introduce macro $Do(\delta, s, g)$, expanding into a formula of the situation calculus augmented by GCATs. It may be read as "*executing the program δ in situation s results in GCAT g*". Note that the last argument of the original Do of Golog and sGolog is respectively a situation and a CAT rather than a GCAT. $Do(\delta, s, g)$ is defined as follows:

$$
\begin{aligned}
&Do(a, s, g) \doteq executable(a, s) \wedge g = a,\ \forall a \text{ primitive action.}\\
&Do(multibranch\_on(v), s, g) \doteq Kref(v, s) \wedge g = [\phi_v, \epsilon, \epsilon].\\
&Do(switch(case(v_1,e_1), case(v_2, e_2)), s, g) \doteq\\
&\qquad\qquad Do((?(case(v, v_1)) ; e_1), s, g) \vee\\
&\qquad\qquad Do((?(case(v, v_2)) ; e_2), s, g).\\
&Do(switch(g_1), s, g_2) \doteq g_1 = \epsilon \wedge g_2 = \epsilon.
\end{aligned}
$$

where $Do(?(case(v, v_i)), s, g)$ holds in case v_i is a potential value for the term v. $Do(\beta, s, g)$ with β being $(\delta_1; \delta_2)$, (δ^*), $(\phi?)$, $(\delta_1|\delta_2)$ or $(proc\ \delta\ endproc)$ is defined as in [103], by simply changing the CAT c by GCAT g.

It is straightforward to prove that the Golog and the s_{sl}Golog interpreter coincide in case we confine ourselves to Golog programs without sensing, and occurrences of the special actions *multibranch_on* and *switch*.

Adapting Backward Chaining for s_{sl}Golog

We suggest four new axioms for regression-based planners to elaborate Web service composition with conditional branches in s_{sl}Golog:

- Axiom 4.19;
- Axiom 4.20;
- Axiom 4.21;
- Axiom 4.22.

First of all, axiom (4.19) is required to introduce a branching node during backward chaining based search. To this end, *branch* denotes what is a true in branches. Even if all branches have same facts A in axiom (4.19), backward chaining can be followed differently in each branch.

$$
\begin{aligned}
&holds(A, do(W, s)) \leftarrow\\
&\qquad holds(A\ branch\ A, do(W, s)) \wedge\\
&\qquad W = (multibranch\_on(X); switch(case(X_1, W_1), case(X_2, W_2)))
\end{aligned}
\tag{4.19}
$$

For any axiom $holds(A, do(a, s)) \leftarrow holds(A', s)$, two further axioms are required:

$$
\begin{aligned}
&holds(A\ branch\ B, do(multibranch\_on(v); switch(case(v_1, w; a), case(v_2, y)), s)) \leftarrow\\
&\qquad holds(A'\ branch\ B, do(multibranch\_on(v); switch(case(v_1, w), case(v_2, y)), s))
\end{aligned}
\tag{4.20}
$$

$$
\begin{aligned}
&holds(B\ branch\ A, do(multibranch\_on(v); switch(case(v_1, x), case(v_2, w; a)), s)) \leftarrow\\
&\qquad holds(B\ branch\ A', do(multibranch\_on(v); switch(case(v_1, x), case(v_2, w)), s))
\end{aligned}
\tag{4.21}
$$

By means of axioms (4.20) and (4.21), successor state axioms can be used along any branch. In addition to the previous axioms each conditional action requires a further successor state axiom. By considering actions such as *AdslEligibility* in Figures 4.5 and 4.8, this axiom is in the form of (4.22).

$$\begin{aligned} &holds(SlowNetworkConnection(nc)\ branch\ FastNetworkConnection(nc),\\ &\qquad do(multibranch\_on(nc)\ ; switch(case(slow,x),(fast,y)),s) \leftarrow\\ &Instance(PhoneNumber, ph\_nb, s) \wedge\\ &Instance(Zipcode, code, s) \wedge\\ &Instance(Email, em, s) \end{aligned} \tag{4.22}$$

Such axioms are required to instantiate variables i) X of the action $multibranch\_on(X)$, ii) X_1 and X_2 of the multi-branching node $switch(case(X_1, W_1), case(X_2, W_2))$ in (4.19).

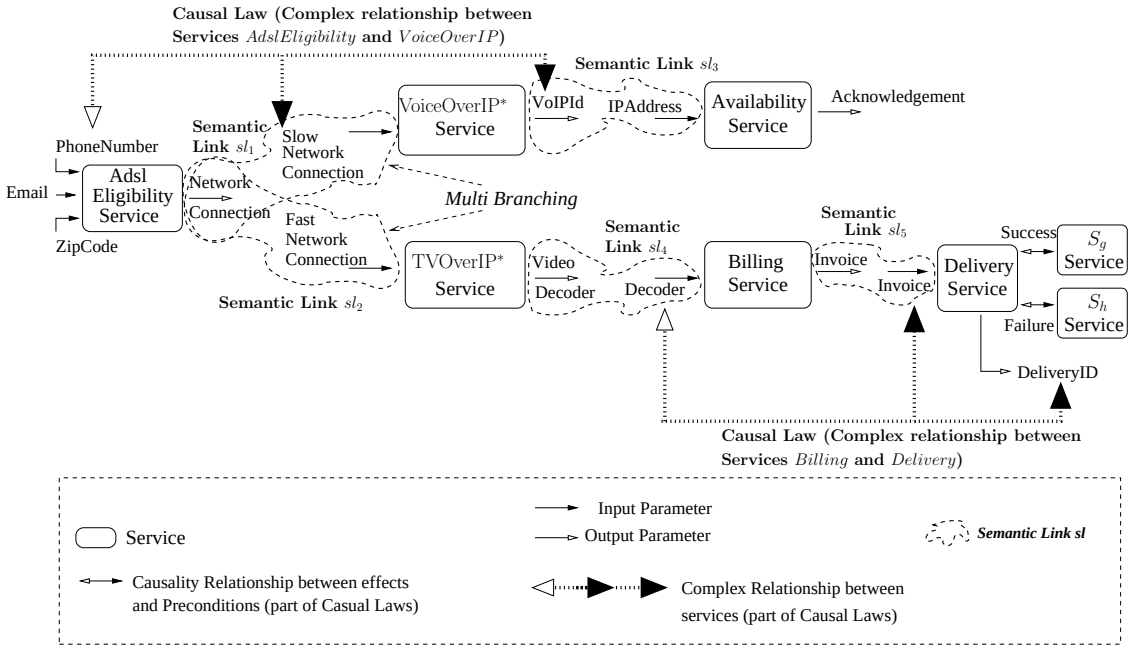


Figure 4.8: A Sample of a Web Service Composition ω_1.

Example 40. *(Backward chaining based search and s_{sl}Golog)*
Here we aim at modelling a subscription process for an Internet Access with TV or Voice over IP. To this end a composition of actions (more actions than in Figure 4.5) are required. Therefore we consider further actions such as `TVOverIP*`, `Availibility`, `Billing`, `Delivery` *illustrated in Tables 4.2 and 4.3. The following s_{sl}Golog program (illustrated in Figure 4.8[8])*

[8]This figure illustrates all semantic links involved in the composition, but only a part of causal laws. On the one hand causality relationships between effects and preconditions of actions are not modelled. On the other hand the complex relationships related `TvOverIP*`, `Billing` services are not illustrated for ease of reading. Details of these causal laws are described in Tables 4.2 and 4.3.

proc ω_1
$\quad$ `AdslEligibility` ; *multibranch_on(NetworkConnection)* ;
$\quad$ *switch([case(slow,* `VoiceOverIP`* ; `Availability`*),*
$\quad\quad$ *case(fast,* `TVOverIP`* ; `Billing` ; `Delivery` ;
$\quad\quad\quad$ *multibranch_on(Delivery_effect);*
$\quad\quad\quad$ *switch([case(success , S_g),*
$\quad\quad\quad\quad$ *case(failure , S_h)])])])*
endproc

has been inferred by means of our extended backward chaining based search (i.e., axioms (4.19)*,* (4.20)*,* (4.21)*, and axiom* (4.22) *for conditional actions). To this end we used AX the foundational axioms of our extended situation calculus.*

In this example we considered the following links and laws to perform Web service composition:

- *semantic links between actions* `AdslEligibility` *and* `VoiceOverIP`* *i.e., axiom* (4.16)*),* `VoiceOverIP`* *and* `Availability`*,* `AdslEligibility` *and* `TVOverIP`**,* `TVOverIP`* *and* `Billing`*,* `Billing` *and* `Delivery`*;*
- *causal laws between actions:*
  - *causality relationships between effects and preconditions of actions (e.g., axiom* (4.10) *in Example 34). Since the actions* `VoiceOverIP`* *and* `TVOverIP`* *require the referent of $NetworkConnection(x)$ to be known, their preconditions axioms are in part defined by $Kref(NetworkConnection(x), s)$;*
  - *explicit and complex relationships between actions (e.g., axiom* (4.9)*).*

The `AdslEligibility` *action is the only binary sensing action such that $sr(AdslEligibility, s)$ be $NetworkConnection((x), s)$ (see Examples 32 and 33).*

The backward chaining consists in applying (4.19)*, twice* (4.20) *for Availability and $VoiceOverIP^*$, twice* (4.21) *for Billing and $TVOverIP^*$, once* (4.22) *and then checking facts in S_0.*

The classic Golog interpreter, running offline, tries to logically derive a linear sequence of primitive actions which are legally executable and representing only an execution trace of ω_1. The selected sequence is only known at runtime after the execution of the sensor $NetworkConnection$, which is the main issue. $s_{sl}Golog$, running offline, builds conditional compositions for any branch points (i.e., conditioned-on fluents) and then for those that do not adhere to the *reasonable persistence* of information. Retrieving such compositions is a real issue in the area of web services, especially because of its dynamic environment.

Executing Web Service Composition

Towards the issue of Web service composition execution, we adapt $s_{sl}Golog$ to find the appropriate terminating GCAT by combining online execution of sensing actions with offline simulation of world-altering actions. To accommodate both backtracking and online sensing we assume that the truth value of a certain fluent, say $F(\vec{x}, s)$ can be determined by executing an external function call (external to $s_{sl}Golog$), a. The call is denoted by $exec(a(\vec{x}), s)$. Whenever the execution succeeds, F is true; otherwise, it is false. Since Prolog answers queries with free variables by returning possible values for these variables, this technique is equally suitable for sensed functional fluents. The use of external function calls with the asumption of *reasonable persistence* of information, allows us to extend the successor state axiom of a fluent $F(\vec{x})$:

$$F(\vec{x}, gdo(a(\vec{x}), s)) \equiv exec(a(\vec{x}), s) \tag{4.23}$$

The set of rules that calls action a externally is also required.

Example 41. *(Interleaving offline and online execution)*
Let ω_1 be the Web service composition computed in example 40. **In this example we assume that `AdslEligibility`, `TVOverIP`*, `Billing` are pure information-providing services whereas `Delivery` is a world-altering Web service**. *In such a configuration and by using AX' i.e., AX with axiom (4.23), we obtain (by random selection of conditional output parameters; here $fast$ for $NetworkConnection$):*

$$AX' \models Do(\omega_1, S_0, g) \text{ with}$$
$$g = AdslEligibility \ . \ TVOverIP^* \ . \ Billing \ . \ Delivery \ .$$
$$[Delivery\_effect \ , [success \ , \alpha][failure \ , \beta]].$$

Branching on $NetworkConnection$ is no more required in the resulting WSC since the selection has been performed during execution of the information-providing action `AdslEligibility`. Only the world-altering Web service `Delivery` required to be executed in the next step.

In details, axioms (4.24) and (4.25) are required for such an execution.

$$Do(multibranch\_on(nc); switch(case(slow, W1), case(fast, W2)), s, nc; G1) \leftarrow$$
$$sr(NetworkConnection, s) = SlowNC \wedge$$
$$Do(W1, do(nc, s), G1) \tag{4.24}$$

$$Do(multibranch\_on(nc); switch(case(slow, W1), case(fast, W2)), s, nc; G2) \leftarrow$$
$$sr(NetworkConnection, s) = FastNC \wedge$$
$$Do(W2, do(nc, s), G2) \tag{4.25}$$

In the same way we have

- $Do(a, s, a)$ *for atomic actions;*
- $Do(a; b, s, a; b) \leftarrow Do(a, s, g_a), Do(b, do(a, s), g_b)$ *for sequences;*

and so on.

Once the conditional compositions are determined, we execute Web services only when needed since the *reasonable persistence* of information is a hard assumption we cannot consider in Telecommunication domain. In case the *reasonable persistence* of information is violated, the composition does not require to be re-planned from scratch but only to be executed online (for services with conditional effects) from the previous conditional composition.

4.2.6 Synthesis

Limitations, Solutions and Extensions

In our work, branching is limited to an enumeration of possible results of a sensing action (i.e., known a priori through semantic description of actions). However, it is still possible to handle branching on more restricted sensed values e.g., a sensing action on the `netSpeed` of a `SlowNetworkConnection` and a branch on `netSpeed`.$256KBS$. An intuitive solution consists in considering a specialized match type for semantic links wherein `SlowNetworkConnection` is involved. For instance we will consider

$$\texttt{SlowNetworkConnection} \sqsupset \texttt{NetworkConnection} \sqcap \texttt{netSpeed}.256KBS \tag{4.26}$$

as one of possible and valid match types between two actions.

The main direction for continuing this work is to combine our work and the approach of [182] to cope first with more large and complex composition, and second with more complex structures like loops and concurrency.

Some Concluding Remarks

In the same direction as Section 4.1 we suggest to study automated (**requirement** $R^{Composition}_{Automation}$) AI planning based Web service composition.

More specially, the approach presented in this section has been directed to meet an interesting challenge facing Web service composition i.e., *how to effectively compute a (conditional) composition of Web services with expressive descriptions* (i.e., services described with input, output parameters and preconditions, effects - compliant with **Requirement** $R^{Service}_{Expressivity}$).

In addition we consider a Web service composition that deals with semantic links (**Requirement** $R^{Semantic}_{Composability}$ through DL reasoning) and causal laws (**Requirement** $R^{Causal}_{Composability}$ through AI planning) between Web services. To this end, we suggested adding explicit DL reasoning (e.g., Subsumption) between input and output parameters of services (semantic links) to the situation calculus. Therefore we refine the definition of executable services to cope with both links and laws. This work presents an augmented version of the logic programming language Golog i.e., s_{sl}Golog (supporting **requirement** $R^{Composition}_{Expressivity}$ with sequential and conditional compositions) that provides a natural formalism for automatically reasoning about the semantic links and causal laws, and so composing services. Such a version of Golog extends sGolog by supporting conditional n-ary parameters of actions to reason about them. Considering conditional plans is obviously flexible and adapted to the open world of Web services.

Finally, online execution of information-providing services with offline simulation of world-altering services are combined to obtain conditional Web service compositions ready for execution.

Even if using conditional plans includes sometimes unrealistically high number of alternative paths as the conditional plans should cover all the possible alternatives, the DL reasoning based semantic link axioms can be used i) to reduce the search space ii) to reduce the number of these alternative plans by retrieving more relevant WSC, iii) to improve performance of WSC (see the experimental evaluation of the proposed approach in Section 7.3.2 of Chapter 7). Thus approaches that consider semantic links and causal laws such as ours, are more restrictive than approaches which consider only semantic links or causal laws.

In comparaison with the work of [139] (see Section 2.1.3), our offline interpreter i) overcomes the assumption related to the reasonable persistence of information, ii) supports information-providing as well as world-altering Web services, iii) provides a strategy for automated branching output parameters (as sensed values) to input parameters by means of DL reasoning and their valid semantic links (Requirement $R^{Semantic}_{Composability}$), iv) couples with DL reasoning and Golog reasoning to deal with expressive description of Web services. Moreover, instead of producing a linear sequence of services (such as many Golog-based approach), our approach computes Web service composition as legal execution of services conditioned on the possible n-ary output parameters of information-providing actions along the way.

4.3 Conclusion

In this chapter we presented two approaches to overcome Web service composition in the semantic Web. Both approaches focus on **requirement** $R^{Composition}_{Automation}$ to achieve composition.

Towards this issue we i) study two levels expressivity for Web services description, ii) consider different levels of composability criteria, iii) model composition with different compositions constructs, the whole by disregarding the hard assumption related to persistence of information. Indeed expressivity of Web services may vary depending on the application domain. Indeed some industrial applications may require to compose Web services which are described by means of input and output parameters (e.g., SA-WSDL based Web service description), or of relationships between actions, preconditions and effects, or of both (e.g., WSMO or OWL-S based Web service description).

The first approach, presented in Section 4.1, couples the matrix of semantic links (i.e., SLM in Section 3.2) and AI planning techniques in order to achieve composition. In such an approach they focus on a part of **Requirement** $R^{Service}_{Expressivity}$ to perform composition i.e., input and output parameter of Web services. In this direction semantic links between Web services are used as composability criteria (**Requirement** $R^{Semantic}_{Composability}$). The result of this composition process is a partial ordering of Web services arranged in a simpler version of a workflow that satisfies **Requirement** $R^{Composition}_{Expressivity}$ by supporting composition constructs such as Sequence, Non Determinism of services, concurrency.

Unlike the approach presented in Section 4.1 that disregards preconditions and effects on parameters of Web services, the second approach (Section 4.2) considers them together with semantic annotation on input and output parameters. This composition approach deals with information-providing and world-altering services as well. This ensures to support the full part of **requirement** $R^{Service}_{Expressivity}$. In addition semantic link ($R^{Semantic}_{Composability}$) and causal laws ($R^{Causal}_{Composability}$) are both regarded as composability criteria. The augmented and adapted version of the logic programming language Golog [121] i.e., s_{sl}Golog has been presented as a natural formalism not only for reasoning about the latter links and laws, but also for automatically composing services. Such a language supports sequential and conditional composition as Requirement $R^{Composition}_{Expressivity}$.

In both approaches, valid and/or robust semantic Web service compositions have been addressed in details.

Table 4.4 describes in details the requirements supported by the models introduced in Chapter 4.

Since our approach has been directed to meet semantic links based Web service composition with causal laws, it becomes conceivable to further exploit semantic links between their functional parameters. More specifically the quality of semantic links (robust or valid) involved in a composition can be then used as a innovative and distinguishing criterion to estimate its overall semantic quality. Therefore, in the following section we will study in more details semantic quality of semantic link based semantic Web service composition.

<table>
<tr><th rowspan="2">Requirement R_i</th><th rowspan="2">Details</th><th colspan="2">Chapter 4</th></tr>
<tr><th>Section 4.1</th><th>Section 4.2</th></tr>
<tr><td rowspan="2">$R^{Composition}_{Automation}$</td><td>Formalism</td><td>Independent</td><td>Situation Calculus</td></tr>
<tr><td>Composition Mechanism</td><td>(Robust) Ra_4C</td><td>Golog + s_{sl}Backward Chaining</td></tr>
<tr><td rowspan="3">$R_{Expressivity}$</td><td rowspan="2">$R^{Service}_{Expressivity}$</td><td colspan="2">In both Sections, input and output parameters are supported to describe Web services. Such parameters</td></tr>
<tr><td>are annotated by concepts using a common domain ontology. Information-providing services.</td><td>Preconditions and effects can be considered to ensure possible and valid compositions of services. Information-providing and World-altering services.</td></tr>
<tr><td>$R^{Composition}_{Expressivity}$</td><td>The supported control constructs are as follows: sequence, non determinism choice of Web services and concurrent compositions.</td><td>The supported control constructs are as follows: sequence and conditional compositions.</td></tr>
<tr><td rowspan="2">$R_{Composability}$</td><td>$R^{Semantic}_{Composability}$</td><td colspan="2">Semantic links ar required in this approach. In particular, their valuation is based on matchmaking functions i.e., Exact, PlugIn, Subsume, Disjoint, Intersection, Abduction and Difference. However this set can be further extended. Computation at Design Time</td></tr>
<tr><td>$R^{Causal}_{Composability}$</td><td>✗</td><td>Causality relationships between effects and preconditions of services. Complex and explicit relationships between services.</td></tr>
<tr><td colspan="2">$R_{Flexibility}$</td><td>Supported by the SLM model.</td><td>Supported by the semantic links and causal law axioms.</td></tr>
<tr><td colspan="2">$R_{Optimization}$</td><td colspan="2">✗</td></tr>
<tr><td rowspan="2">$R_{Applicability}$</td><td rowspan="2">$R^{Service}_{Applicability}$</td><td colspan="2">Applicable to the OWL-S service profile, WSMO</td></tr>
<tr><td>SA-WSDL specification.</td><td>service capability SWSO Inputs/Outputs.</td></tr>
<tr><td>$R^{Composition}_{Applicability}$</td><td colspan="2">✗</td></tr>
</table>

Table 4.4: Table of Requirements supported by Chapter 4. Legend: ✗ = not addressed.

Chapter 5

Optimizing Semantic Link based Web Service Composition

Contrary to Chapter 4 which considers Web service composition as a *process*, here we consider and study the *result* of the composition process. This result is also known as the *composite Web service*. Throughout the Chapter, *composition* and *composite Web service* are used synonymously.

According to Chapter 4, composite Web services can be computed by considering:

- semantic links between functional output and input parameters of their Web services;
- and causal laws.

as composability criteria.

Even if the approach presented in Section 4.2 is appropriate to constrain and restrict the set of service compositions by extending the definition of an *executable* service (i.e., *possible* and *valid* in Equation (4.12)), there are still many (e.g., more than one) candidate compositions that can be returned by this approach, and a fortiori by approach presented in Section 4.1 (which is less restrictive).

In some cases, some of these candidate composite services are "*unsuitable*" to achieve a given goal since most of their semantic links are not robust (see Definition 13). Such composite services which do not provide acceptable qualities[1] of semantic links might be as useless as services that do not provide the desired functionality.

Towards these issues

i) of further pruning the set of numerous candidates;

ii) and of selecting the most *suitable* compositions,

we suggest to exploit the quality of semantic links (through their estimation and ranking; see Table 3.1) involved in the service compositions.

[1] As previously said in Chapter 3 this quality is valued by matching functions such as Exact, PlugIn, Subsume, Intersection and Disjoint.

This chapter is then interesting in computing semantic link based *optimal* Web service composition, hence satisfying the **requirement** $R_{Optimization}$. Starting from an initial set of web services, the goal of this chapter aims at

> *selecting web services and maximizing the overall quality of their inter-connections by means of their semantic links according to a goal to achieve. Such a selection is achieved by taking into account preferences and constraints defined by the end-user.*

Unlike most approaches [28, 206, 208] which focus on the quality of composition by means of non functional criteria such as the quality of service (QoS), here the quality of semantic links is considered. Such a criterion can be viewed as an innovative and distinguishing functional criterion to estimate the overall semantic quality of web service compositions. By considering such a new criterion, we extend the family of criterion that can be used to value different Web service compositions satisfying the same goal. The problem of optimization in service composition is addressed with respect to this functional criterion.

The remainder of this chapter is organised as follows. In Section 5.1 we briefly sketch the main required background of this chapter and review in more details the semantic link composition model we will focus on. Section 5.2 presents a general and extensible model to evaluate and estimate quality of both elementary and semantic links based Web service composition. Section 5.3 formulates the problem of the semantic link based *optimal* Web service composition and focuses on three approaches to solve it i.e., i) a local[2], ii) a naive and iii) a global selection approach. The local and naive approaches have been introduced to compare the characteristics of their optimal compositions and also their computation time performance with solutions of our global selection approach. The latter method is described through an integer linear programming approach that efficiently solves the optimization problem of selection. Finally section 5.4 draws some conclusions and talks about possible future directions.

5.1 Background

In this section we first briefly sketch the main background of the chapter i.e., *semantic link based Web service composition* (Section 4.1), and *(robust) semantic links*. Then, we *model* in further details the semantic link based Web service composition at *different levels of abstraction*.

5.1.1 Web Service Composition, Semantic Links and Robustness

In this chapter we assume that a non empty set of Web service compositions have been discovered to achieve a target goal. Such a set of compositions have been computed by considering i) semantic links (Requirement $R^{Semantic}_{Composability}$), or ii) semantic links (Requirement $R^{Semantic}_{Composability}$) and causal laws (Requirement $R^{Causal}_{Composability}$) as composability criteria. In other words, the method presented in this chapter can re-use compositions resulting of approaches which are presented in Sections 4.1 and 4.2 of Chapter 4.

Since we plan to achieve **semantic link** based *optimal* Web service composition (Requirement $R_{Optimization}$), Chapter 3 is the main required knowledge for the understanding of this Chapter. In more details, we required knowledge about the following main definitions, theorem and properties of Chapter 3:

[2]By local approaches, we refer to approaches in which the semantic links are selected individually and locally in the composite service

- the *Semantic Links* Definition (Definition 10 that formalizes inter-connections between Web services; see an illustration in Figure 3.1);
- the *Matching Types* Definition: *Exact* ($\equiv$), *PlugIn* ($\sqsubseteq$), *Subsume*($\sqsupseteq$), *Intersection* ($\sqcap$) and *Disjoint* ($\perp$) (Section 3.1.2) which are employed to value semantic link between Web services. The partial order (Theorem 1) of the latter *Matching Types* as well as their valuation (discretization of match types in Table 3.1) are required in this Chapter;
- the *properties* on Semantic Links, which depends on the match type used to value them i.e., Validity (Definition 11), Robustness (Definition 13);
- the *properties* on *semantic links* based Web service composition i.e., Validity (Definition 12), Robustness (Definition 14).

Such properties and definitions can be used together to semantically value and compare not only different individual semantic links but also different Web service compositions satisfying the same goal. For instance, these compositions can be (semantically) compared by means of the following criteria (ordered by computational time complexity):

- an overall *Matching Quality* of semantic links;
- a degree of *Validity*;
- a degree of *Robustness*;
- the *Common Description rate* which is defined as the rate of common description in the two parameters of the semantic links;
- the *Extra Description rate* which is defined as the rate of extra description required to change a semantic link into its robust form.

In the following, we will focus on a specific subset of these criteria i.e. *Matching Quality*, *Robustness* and *Common Description rate* to value semantic links and their composition. Roughly speaking the *Robustness* criterion has been preferred to the *Validity* criterion since the latter criterion is too restrictive to value semantic links. The *Common Description rate* can be changed with the *Extra Description rate* since these two rates are complementary. The higher *Extra Description rate* the lower *Common Description rate* and vice versa.

5.1.2 Modelling Semantic Link Based Web Services Composition

In this section we first briefly remind main results about composite service specifications, and more specifically the constructs which are used to model these (expressive) compositions (Requirement $R^{Composition}_{Expressivity}$). Then we describe two different levels of abstraction to model compositions i.e., the *abstract* (or generic) and *practical* levels. These two levels of composition will be used along this Chapter to present our approach.

Composite Service Specifications

In this chapter and in the same way as in Chapter 4 (see Sections 4.1 and 4.2), the process model of Web service compositions and their semantic links are graphically specified by means of a statechart [83].

This choice has been motivated by several reasons. First, statecharts possess a formal semantics, which is essential for analyzing composite service specifications. Second, statecharts are a well-known and well-supported behaviour modelling notation. Finally, statecharts offer most of the control-flow constructs found in existing process modelling languages and they have been shown to be suitable for expressing typical control-flow dependencies. Hence, it is possible to adapt the semantic based service selection mechanisms developed using statecharts to fit other alternative languages.

In more details, *states* of a statechart refer to Web services. *Transitions* are labelled with semantic links between Web services. In addition some basic composition flow constructs such as sequence (i.e., linear compositions), conditional branching (i.e., OR-Branching), concurrent threads (i.e., AND-Branching) can be found[3]. To simplify the presentation and align our work with composition constructs introduced by Definition 19 and supported by our approaches of Sections 4.1 and 4.2, we initially assume that all considered statecharts have no structured loops (acyclic[4]), or inter-thread synchronization.

In case a composition (through its graphical representation i.e., statechart) contains cycles, a technique for *unfolding* it into its acyclic form needs to be applied beforehand. Thus such cyclic compositions can be represented by a sequential flow by unfolding the cycles. Roughly speaking our approach is similar to what is proposed by [209]. If a composition contains cycles, these need to be "*unfolded*" so that the resulting composition has a finite number of practical compositions and semantic links. The method used to unfold a composition is to examine the logs of past executions to determine the maximum number of times that each cycle is taken. The states appearing between the beginning and end of a cycle are then cloned as many times as the transition causing the cycle is taken[5]. This unfolding method works if the beginning and end of each cycle in the composition can be identified.

Generic Service Tasks, Abstract Semantic Links and their Abstract Composition

Here, we focus on an *abstract* (or generic) level to model compositions. Such compositions are defined by a composition of *generic service tasks*.

Definition 23. ***(Generic Service Tasks)***
Generic service tasks are semantically defined in the same way as semantic Web services (i.e., input, output parameters and preconditions, effects), but in an abstract way.

The Definition 23 means that a non pre-defined Web service is selected to achieve the given generic service task. The generic service task is then abstracted since it is not instantiated.

Example 42. ***(Illustration of a Generic Service Task)***
Figure 5.1 illustrates a generic service task. Such a task $\texttt{VoiceOverIP}^T$ *starts from a* `NetworkConnection` *and a* `Phone Number`*, returns the* `Address` *of the ADSL line a Telecom operator needs to install the line. Contrary to the service* `VoiceOverIP`*, the generic service task* $\texttt{VoiceOverIP}^T$ *cannot be executed since no implementation has been discovered.*

In the following we will use the term **Tasks** to refer to **Generic Service Tasks**.

[3]These constructs are considered, in our Ph.D thesis, as essential to model Web service composition.
[4]This is the case in our thesis since the compositions of Web services computed in Chapter 4 are devoid of cycles.
[5]The idea of cloning the states in a loop for the purpose of transforming a cyclic statechart into an acyclic one has been proposed by [79]

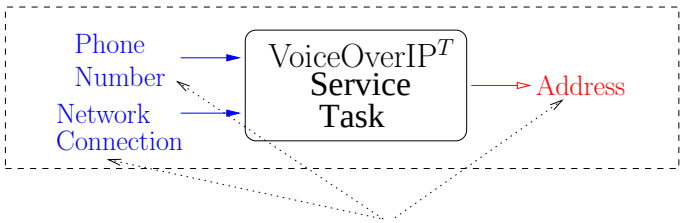


Figure 5.1: Illustration of a Generic Service Task.

A composition of the latter tasks, so called *abstract composition* is defined in the same way as semantic link based Web service composition. Indeed *abstract compositions* have *abstract semantic links* between their tasks whereas Web service compositions have *semantic links* between their Web services. The main difference is about the term used to formalize their *links*. The meaning of these links is the same. Therefore abstract semantic links $sl^A_{i,j}$ (here between Task T_i and Task T_j) are used in an abstract composition specification to capture the data manipulation perspective. Specifically, they can be used to express branching conditions.

Definition 24 is used to formalize an *abstract composition*.

Definition 24. ***(Abstract Composition)***
An abstract composition is specified as a collection of tasks described in terms of service ontologies and combined according to a set of control-flow and abstract semantic link (or data-flow) dependencies.

Remark 7. ***(Abstract Composition Specifications)***
In the same way as Web service composition, the process model of abstract compositions and their abstract semantic links are graphically specified by means of a statechart.

In more details, states *of a statechart refer to tasks.* Transitions *are labelled with abstract semantic links between tasks.*

Example 43. ***(Process Model of an Abstract Composition)***
Suppose $T_{i,1\leq i\leq 8}$ be eight tasks involved in an abstract composition. The process model of this abstract composition is illustrated in Figure 5.2. The abstract composition consists in a sequence of tasks intertwined of an OR-Branching and an AND-Branching wherein nine abstract semantic links are involved. In other words, given the conditional output parameter of task T_1, either the latter output parameter is bound to the input parameter of T_2 or it is bound to the input parameter of T_4. Therefore two potential and conditional compositions are possible i.e., one that contains T_2 and another one that contains T_4. From the task T_5, its output parameter is semantically linked an input of tasks T_6 and T_7 i.e., T_6 and T_7 are done concurrently.

From this we define *practical composition* which is a concretization of an *abstract composition*.

Web Services, Semantic Links and their Practical Composition

Here, we focus on a *practical* level to model compositions. Such compositions are defined by a composition of *Web services*.

An Abstract composition can become *practical* in case we discover a non empty set of candidate Web services that can execute each task of the abstract composition.

To this end any Web service s_i that can execute T_i have to meet the following requirements. First of all s_i achieves the same overall functionality defined by T_i. Secondly, any functional input

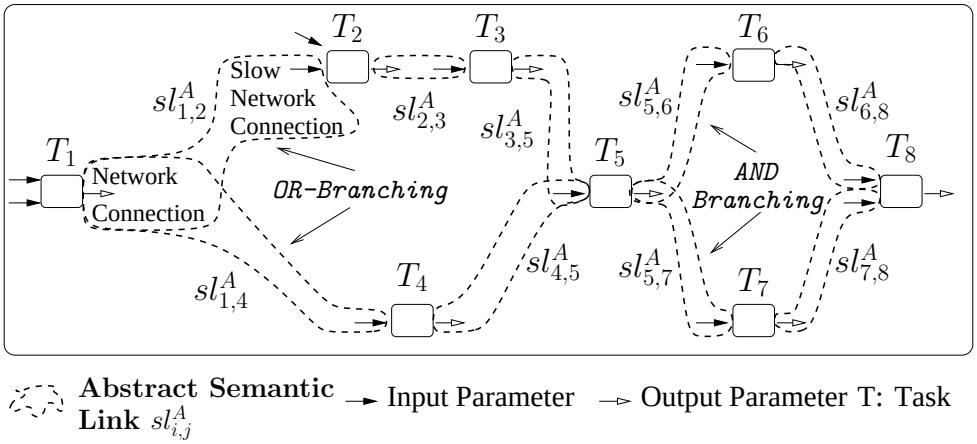


Figure 5.2: Illustration of an Abstract Composition.

(output) parameters of T_i has semantic similarity (i.e., a satisfiable conjunction) with one of the s_i's input (output) parameter. Finally, the service s_i has exactly the same preconditions and effects as task T_i. Therefore a collection of Web services that meet i) the (overall) functionality and ii) the semantic description of a given task can be selected to achieve a given task of an abstract composition.

Example 44. ***(Task and a Candidate Web Services)***
According to the previous definition, it is obvious that service `VoiceOverIP` *(illustrated in Section 1.3.1 and* s_2 *in Figure 5.3) is a candidate to achieve task* $\texttt{VoiceOverIP}^T$ *in Example 42. Indeed they have the same overall functionality i.e., computing an address given a network connection and a phone number. Both service and task have same preconditions and effects. Moreover there are semantic relationships between parameters of the service and task (see Figure 5.3) i.e.,*

- *they have the same semantic description for the input parameter* `Phone Number` *i.e.,* Exact*;*
- *the input parameter* `SlowNetworkConnection` *of* `VoiceOverIP` *(*s_2*) is subsumed by* `NetworkConnection` *i.e., the input parameter of task* $\texttt{VoiceOverIP}^T$*;*
- *the output parameter* `Address` *of* $\texttt{VoiceOverIP}^T$ *subsumes the output parameter* `VoIPId` *of* `VoiceOverIP`*.*

In the previous example we focused on a unique candidate Web service to achieve a task. However there are many cases wherein a collection of candidate services could achieve a same task. Indeed some services with common functionality, preconditions and effects although different input and output parameters can be used to achieve a target task in the abstract composition. For instance we can imagine many different levels (but semantically close) of functional description to model tasks that achieve a same functionality in the open world of Web services.

Example 45. ***(Illustration of a Task and its Collection of Candidate Web Service)***
Figure 5.3 illustrates a collection of services that are candidates to achieve task $\texttt{VoiceOverIP}^T$ *i.e.,* $\{s_2, s'_2, s''_2\}$.

The concept of *Practical Composition* defined below captures the various ways of performing a given abstract composition.

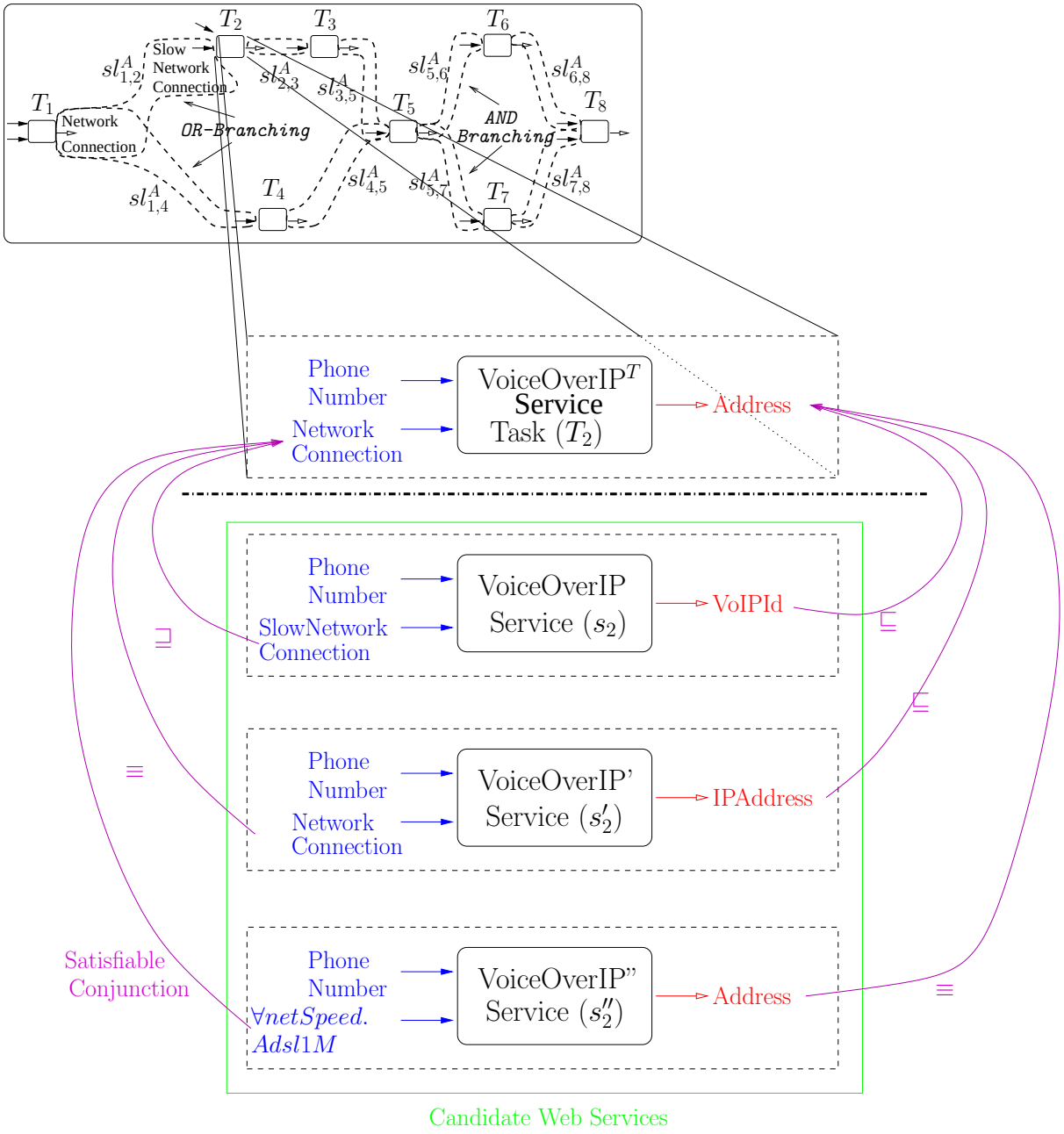


Figure 5.3: Illustration of a Task and its Collection of Candidate Web Services.

Definition 25. ***(Practical Composition)***
A set of pairs $c = \{< T_1, s_1 >, < T_2, s_2 >, ..., < T_n, s_n >\}$ *is a* practical composition *of an* abstract composition c^A *iff:*

- $\{T_1, T_2, T_n\}$ *is the set of tasks in* c^A.

- *For each pair $< T_i, s_i >$ in c, service s_i belongs to the service class associated with task T_i. In other words, service s_i provides operation semantically close to the operation required by task T_i.*

The abstract semantic links of an abstract composition are instantiated by semantic link of a practical composition.

It should be noted that, according to definition 25, each task can be executed by a number of alternative Web services, but it is not possible for a Web service to execute a combination of tasks in a "single shot". To express that a combination of tasks can be executed by a single Web service, these tasks need to be assembled into a single one.

Remark 8. ***(Practical Composition Specifications)***
In the same way as Web service composition, the process model of practical compositions and their semantic links are graphically specified by means of a statechart.

In more details, states *of a statechart refer to Web services.* Transitions *are labelled with semantic links between tasks.*

Example 46. ***(Process Model of a Practical Composition)***
Suppose Example 43 and its abstract composition illustrated in Figure 5.2. Figure 5.4 illustrates one of its practical composition[6]*. Web services $s_{i,1\leq i\leq 8}$ are selected candidate services to achieve respectively tasks $T_{i,1\leq i\leq 8}$ in the abstract composition.*

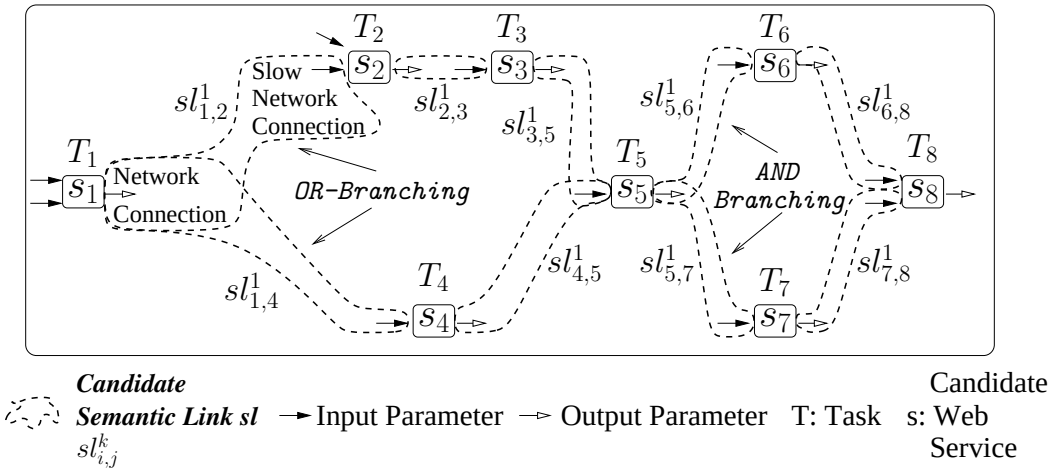


Figure 5.4: Illustration of a Practical Composition.

The example 46 illustrates a practical composition wherein $s_{i,1\leq i\leq 8}$ are involved. More specifically we obtained a practical composition wherein tasks T_i have been concretized by one of their candidate Web services e.g., here s_i for each task T_i. Since a task can be achieved by more than one Web service, they have a large number of potential practical composition that can achieve the same goal. Indeed, it is possible to obtain different practical composition in different ways by allocating different Web services to the basic tasks in the abstract composition.

[6]Here, we intentionally illustrate a process model of a composition with Web services of Section 1.3.1. The introduced composition is a particular case of the composition computed in Example 24 and illustrated in Figure 4.3. Indeed s_1 is `AdslEligibility`, and s_2 is `VoiceOverIP`.

In the following we address the issue of composing a large and changing collection of semantic web services. In our approach Web services belonging to the required tasks are selected at composition time, only based on their semantic links with other Web services. Thus each abstract semantic link $sl^A_{i,j}$ between two tasks T_i, T_j of an abstract composition needs to be concretized. Ideally, a relevant link is selected among its n candidate semantic links $sl^{k,1\leq k\leq n}_{i,j}$ between two of their services to obtain a practical composition.

Example 47. ***(Tasks, Candidate Web Services and Semantic Links)*** *Suppose the semantic link $sl_{1,2}$ between Tasks T_1 and T_2 of the practical composition in example 46 (Figure 5.4). Moreover we assume that T_1 can be achieved by the candidate Web service s_1 i.e., `AdslElegibility`; T_2 by candidate Web services s_2 i.e., `VoiceOverIP`, s'_2 i.e., `VoiceOverIP'`, s''_2 i.e., `VoiceOverIP"` (Figure 5.3). Therefore the tasks T_1 and T_2 can be related by three candidate semantic links (Figure 5.5) i.e.,*

i) $sl^1_{1,2}$ between s_1 and s_2;

ii) $sl^2_{1,2}$ between s_1 and s'_2;

iii) $sl^3_{1,2}$ between s_1 and s''_2.

The semantic link $sl^2_{1,2}$ is more robust than $sl^1_{1,2}$ and $sl^3_{1,2}$. Indeed $sl^2_{1,2}$ is valued by an Exact matching type whereas $sl^1_{1,2}$ is valued by a Subsume matching type, and $sl^3_{1,2}$ is valued by an Intersection match type.

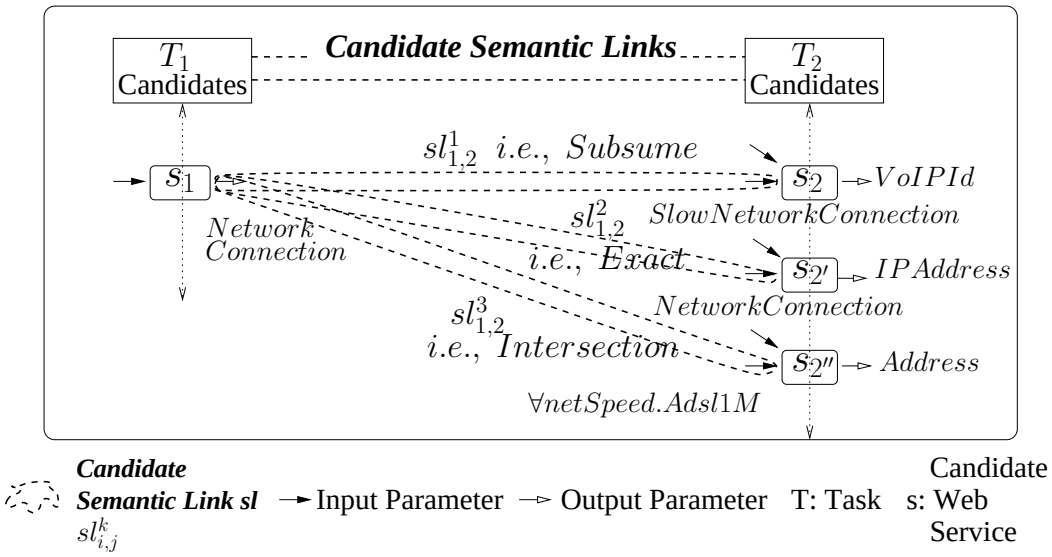


Figure 5.5: Candidate Semantic Links between Tasks T_1 and T_2.

Depending on the selected Web services to achieve tasks, the latter tasks can be related by different semantic links, which are differently valued (e.g., through their robustness, matching quality). In the following we introduce a quality model to value semantic link based Web service composition.

5.2 Semantic Link Quality Model

In the composition model presented in the previous section, candidate services are typically grouped together in tasks of an abstract composition. Different semantic links can be used between tasks, depending on the selected Web services. A way to differentiate their semantic links (e.g., $sl^1_{1,2}$, $sl^2_{1,2}$ and $sl^3_{1,2}$ in Example 47) consists in considering their different functional properties. For this purpose, we adopt a semantic link quality model based on a set of quality functional criteria that are applicable to any semantic link, for example, their robustness.

In this section, we first present the quality criteria in the context of elementary semantic links, before turning our attention to composite semantic links through aggregation functions for sequential, AND-Branching and OR-Branching compositions. For each criterion, we provide a definition, provide rules to compute its value for a given semantic link and we indicate motivations. Finally we study the partial independence of the different quality criteria and their aggregation functions.

5.2.1 Quality Criteria for Elementary Semantic Links

We consider three generic quality criteria for elementary semantic links $sl^k_{i,j}$ defined by $\langle s_i, Sim_{\mathcal{T}}(Out\_s_i, In\_s_j), s_j\rangle^k$: its i) *Robustness*, ii) *Common Description* rate, and iii) *Matching Quality*.

i) **Robustness.**
Given a semantic link $sl^k_{i,j}$ between two Web services s_i and s_j, the Robustness q_r of $sl^k_{i,j}$ is equal to 1 in case the link $sl^k_{i,j}$ is robust (see Definition 13), and 0 otherwise.

Example 48. ***(Robustness)***
Suppose Example 47 and its graphical representation in Figure 5.5. According to the definition of robustness quality, we have:

- $q_r(sl^1_{1,2}) = 0$ *since* $sl^1_{1,2}$ *is not a robust semantic link;*
- $q_r(sl^2_{1,2}) = 1$ *since* $sl^2_{1,2}$ *is a robust semantic link;*
- $q_r(sl^3_{1,2}) = 0$ *since* $sl^1_{1,2}$ *is not a robust semantic link.*

As presented in Chapter 3 such a feature is key to ensure the robustness of a link in a overall composition. Here, our system advertises the robustness of semantic links by pre-computing them.

ii) **Common Description rate.**
Given a semantic link $sl^k_{i,j}$ between s_i and s_j, the Common Description rate $q_{cd} \in (0, 1]$ measures "a" degree of similarity between an output parameter of s_i and an input parameter of s_j. This rate is computed using the following expression:

$$q_{cd}(sl^k_{i,j}) = \frac{|Out\_s_i \sqcap In\_s_j|}{|In\_s_j \backslash Out\_s_i| \ + \ |Out\_s_i \sqcap In\_s_j|} \tag{5.1}$$

where expressions in between | refer to the size of $\mathcal{ALE}$ concept descriptions ([102] p.17) i.e., $|\top|$, $|\bot|$, $|A|$, $|\neg A|$ and $|\exists r|$ is 1; $|C \sqcap D| \doteq |C| + |D|$; $|\forall r.C|$ and $|\exists r.C|$ is $1 + |C|$. For instance $|Adsl1M|$ is 3 in ontology illustrated in Figure 1.4.

Example 49. ***(Common Description rate)***
We still assume Example 47 and its Figure 5.5. According to the definition of common

description rate, we obtain for $sl^1_{1,2}$

$$\begin{aligned} q_{cd}(sl^1_{1,2}) &= \frac{|NetworkConnection \sqcap SlowNC|}{|SlowNC \backslash NetworkConnection| \ + \ |NetworkConnection \sqcap SlowNC|} \\ &= \frac{|\forall netPro.Provider \ \sqcap \ \forall netSpeed.Speed|}{|\forall netSpeed.Adsl1M| \ + \ |\forall netPro.Provider \ \sqcap \ \forall netSpeed.Speed|} \\ &= \frac{(1+1)+(1+(1+1))}{(1+(1+(1+1))) \ + \ (1+1)+(1+(1+1))} \\ &= \frac{5}{9} \end{aligned} \tag{5.2}$$

By performing the same operation on $sl^2_{1,2}$*, we have* $q_{cd}(sl^2_{1,2}) = 1$.
In the same way we obtain for $sl^3_{1,2}$*:*

$$\begin{aligned} q_{cd}(sl^3_{1,2}) &= \frac{|NetworkConnection \sqcap \forall netSpeed.Adsl1M|}{|\forall netSpeed.Adsl1M \backslash NetworkConnection| \ + \ |NetworkConnection \sqcap \forall netSpeed.Adsl1M|} \\ &= \frac{|\forall netSpeed.Speed|}{|\forall netSpeed.Adsl1M| \ + \ |\forall netSpeed.Speed|} \\ &= \frac{(1+(1+1))}{(1+(1+(1+1))) \ + \ (1+(1+1))} \\ &= \frac{3}{7} \end{aligned} \tag{5.3}$$

The presented criterion estimates the common description rate which is well specified for upgrading a non robust semantic link into its robust form. Indeed, in case a semantic link is not robust enough, it seems important to distinguish non robust semantic links by valuing the degree of robustness of these links by a non binary criterion. This is achieved by computing the rate of common description. In equation (5.1), $Out\_s_i \sqcap In\_s_j$ is supposed to be satisfiable since only relevant links between two services are considered in our model. In the same way as the Robustness, the latter rate is provided on requests by the system.

iii) **Matching Quality.**
The Matching Quality q_m of a semantic link $sl^k_{i,j}$ is a value in $(0, 1]$ defined by $Sim_{\mathcal{T}}(Out\_s_i, In\_s_j)$ i.e., either 1 (Exact), $\frac{3}{4}$ (PlugIn), $\frac{1}{2}$ (Subsume) or $\frac{1}{4}$ (Intersection). The Disjoint match type is not considered since it refers to irrelevant semantic links between two services in a composition. Therefore $Out\_s_i \sqcap In\_s_j$ is supposed to be satisfiable.

Example 50. ***(Matching Quality)***
According to the Example 47, Figure 5.5 and the definition of matching quality, we have:

- $q_m(sl^1_{1,2}) = \frac{1}{2}$;
- $q_m(sl^2_{1,2}) = 1$;
- $q_m(sl^3_{1,2}) = \frac{1}{4}$.

Contrary to the common description rate, the matching quality does not estimate similarity between functional parameters of semantic links but gives an general overview of their semantic relationships. Indeed the degree of similarity is, here, more general and indexed on discretized values. By introducing such a criterion, we are motivated towards another level of semantic valuation. In the same way as the Robustness and the common description rate, our system advertises the matching quality of semantic links by pre-computing them.

Abstract Semantic Link $sl^A_{i,j}$		$sl^A_{1,2}$			$sl^A_{1,4}$	$sl^A_{2,3}$			$sl^A_{3,5}$	$sl^A_{4,5}$	$sl^A_{5,6}$	$sl^A_{5,7}$	$sl^A_{6,8}$	$sl^A_{7,8}$
Candidates		$sl^1_{1,2}$	$sl^2_{1,2}$	$sl^3_{1,2}$	$sl^1_{1,4}$	$sl^1_{2,3}$	$sl^2_{2,3}$	$sl^3_{2,3}$	$sl^1_{3,5}$	$sl^1_{4,5}$	$sl^1_{5,6}$	$sl^1_{5,7}$	$sl^1_{6,8}$	$sl^1_{7,8}$
Quality Vector $q(sl^k_{i,j})$	q_r	0	1	0	0	1	0	1	1	0	0	1	0	1
	q_{cd}	$\frac{5}{9}$	1	$\frac{3}{7}$	$\frac{5}{9}$	1	$\frac{3}{5}$	$\frac{3}{4}$	1	$\frac{6}{7}$	$\frac{3}{5}$	$\frac{2}{7}$	$\frac{1}{5}$	$\frac{3}{4}$
	q_m	$\frac{1}{2}$	1	$\frac{1}{4}$	$\frac{1}{2}$	1	$\frac{1}{4}$	$\frac{3}{4}$	1	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{3}{4}$	$\frac{1}{4}$	$\frac{3}{4}$

Table 5.1: Quality Values of Candidate Semantic Links.

Remark 9. ***(Extension of the Semantic Link Quality Model with Contraction [50])*** *In case we consider $Out\_s_i \sqcap In\_s_j$ to be not satisfiable, it is straightforward to extend and adapt our quality model by computing contraction [50] between Out_s_i and In_s_j. Thus, the three quality criteria can be updated in consequence.*

Given the above quality criteria, the quality vector of a semantic link $sl^k_{i,j}$ is defined as follows:

$$q(sl^k_{i,j}) = \big(q_r(sl^k_{i,j}), q_{cd}(sl^k_{i,j}), q_m(sl^k_{i,j})\big) \tag{5.4}$$

Example 51. ***(Abstract Semantic Links and Some of their Candidates)*** *Table 5.1 illustrates the quality values of different candidate semantic links we consider in this Section.*

Even if we illustrated the latter quality criteria by a unique semantic link between Web services, the generalization to n semantic links is straightforward. In case of services s_i and s_j related by more than one semantic link,

- the value of **Robustness** is established by choosing the *minimum* of robustness values of the semantic links involved between s_i and s_j. Indeed a robust composition of Web services is defined as a composition of Web services wherein all its semantic links are robust (Definition 14);
- the overall value of **Common Description rate** is elaborated by computing their *average*. Indeed we assume that the common description rate of all semantic links between two different services have same importance. Therefore we can sketch the average rate of common description the target inter-connection between two services has for each of these semantic links. Note that the *average* can be changed by a *weighted average* as well.
- the **Matching Quality** of several semantic links between two services is defined as their *product* since we would like to easily identify compositions with low or high number of semantic links together with their matching quality. The more semantic links in the composition the highest the probability to obtain relevant inter-connection between services.

Assume s_i and s_j be two Web services where n semantic links $sl_{(i,j)^u, 1\leq u\leq n}$ are involved between them (Figure 5.6). The quality vector $q(sl_{i,j})$ of the general (aggregation of the $sl_{(i,j)^u}$) semantic link between s_i and s_j is defined by (5.5)[7].

$$q(sl_{i,j}) = \big(\min_{1\leq u\leq n} q_r(sl_{(i,j)^u}), \frac{1}{n}\sum_{u=1}^{n} q_{cd}(sl_{(i,j)^u}), \prod_{u=1}^{n} q_m(sl_{(i,j)^u})\big) \tag{5.5}$$

[7]The choice of the minimum on *robustness*, average on *common description rate* and product on *matching quality* is based on different heuristics.

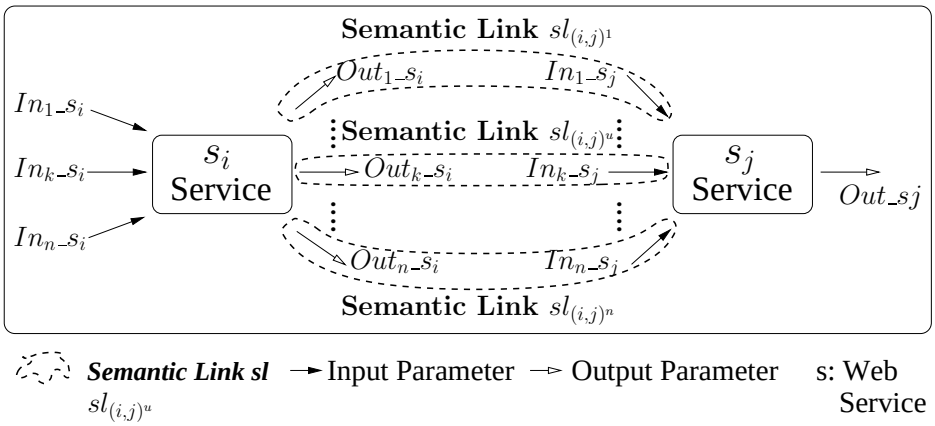


Figure 5.6: Multi-(Semantically) Linked Web Services.

Note that the method for computing the value of the quality criteria is not unique e.g., valuation of Match level. Other computation methods can be designed to fit the needs of specific applications.

Although the adopted quality model has a limited number of criteria (for the sake of illustration), it is extensible: new functional criteria can be added without fundamentally altering the semantic link selection techniques built on top of the model. For instance the *Common Description rate* q_{cd} can be changed by the *Extra Description* rate q_{ed} in (5.6) i.e., the rate of description missing to upgrade the semantic links in their robust forms.

$$q_{ed}(sl^k_{i,j}) = \frac{|In\_s_j \backslash Out\_s_i|}{|In\_s_j \backslash Out\_s_i| \ + \ |Out\_s_i \sqcap In\_s_j|} \tag{5.6}$$

Moreover, it is also possible to extend the quality model to integrate non functional service characteristics such as those proposed by [150].

The global semantic link selection presented in Section 5.3 is independent of these computation methods.

5.2.2 Quality Criteria for Semantic Link Compositions

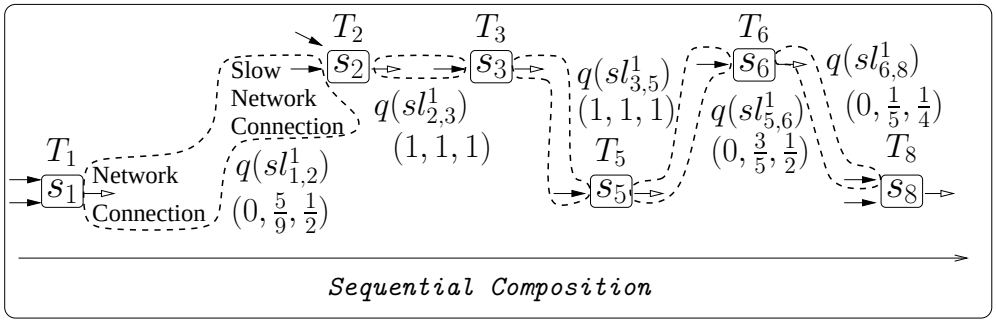
The quality criteria defined above in the context of elementary Web services, are also used to evaluate the quality of any practical composition c of semantic links

$$c = \{< T_1, s_1 >, < T_2, s_2 >, ..., < T_n, s_n >\} \tag{5.7}$$

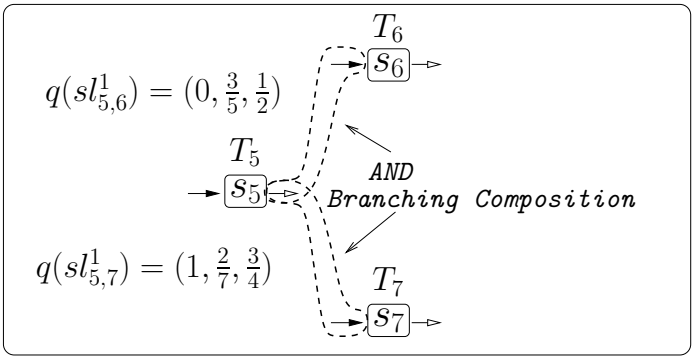
They are computed by aggregation functions. In the following we illustrate the quality criteria on the practical composition c^1, which is defined by

$$c^1 = \{< T_1, s_1 >, < T_2, s_2 >, < T_3, s_3 >, < T_4, s_4 >, < T_5, s_5 >, \\ < T_6, s_6 >, < T_7, s_7 >, < T_8, s_8 >\} \tag{5.8}$$

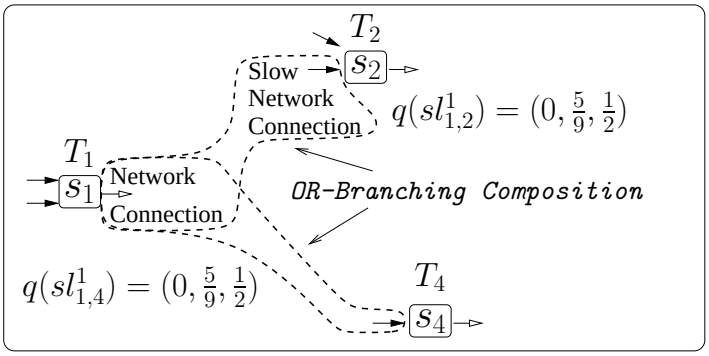
Implicitly the composition c^1 consists of $sl^1_{1,2}$, $sl^1_{1,4}$, $sl^1_{2,3}$, $sl^1_{3,5}$, $sl^1_{4,5}$, $sl^1_{5,6}$, $sl^1_{5,7}$, $sl^1_{6,8}$ and $sl^1_{7,8}$ semantic links.

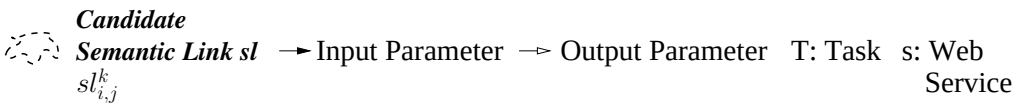


(a) Extract of a Sequential composition $c^1_{Sequence}$.



(b) Extract of an AND-Branching composition c^1_{AND}.





(c) Extract of an OR-Branching composition c^1_{OR}.

Figure 5.7: Pure Sequential, AND-Branching and OR-Branching Compositions extracted from c^1.

Since we focus on practical composition of semantic Web service, we extend the previous quality criteria for such compositions. In this direction Table 5.2 summarizes aggregation functions to evaluate composition. An explanation, formalization, motivation and illustration[8] of each criterion's aggregation function follows:

i) **Robustness.**

- On the one hand the robustness Q_r of both a sequential $c_{Sequence}$ and an AND-Branching composition c_{AND} is defined as the *minimum* of the semantic links $sl^k_{i,j}$'s robustness $q_r(sl^k_{i,j})$ involved in the sequential $c_{Sequence}$ or AND-Branching c_{AND} composition.

$$\begin{aligned} Q_r(c_{Sequence}) &= Q_r(c_{AND}) \\ &= \min_{sl^k_{i,j}}\{q_r(sl^k_{i,j})\} \end{aligned} \tag{5.9}$$

  Indeed a robust composition of Web services is defined as a composition of Web services wherein all its semantic link are robust (Definition 14).

- On the other hand the robustness of an OR-Branching semantic link composition c_{OR} is a *sum of* $q_r(sl^k_{i,j})$ *weighted by* $p_{sl^k_{i,j}}$.

$$Q_r(c_{OR}) = \sum_{sl^k_{i,j}} q_r(sl^k_{i,j}).p_{sl_{i,j}} \tag{5.10}$$

  Therefore each conditional branch is weighted by $p_{sl^k_{i,j}}$ i.e., the *probability that the semantic link* $sl^k_{i,j}$ *be chosen at run time*. This probability is inferred from logs of previous computed compositions. This ensures to obtain a linear function of the latter qualities wherein conditional OR-branches c_{OR} are weighted.

Example 52. ***(Robustness of Sequential and AND-Branching Composition)*** *Let* $c^1_{Sequence}$ *be the sequential composition. Such a composition extracted from* c^1 *is defined by* $\{< T_1, s_1 >, < T_2, s_2 >, < T_3, s_3 >, < T_5, s_5 >, < T_6, s_6 >, < T_8, s_8 >\}$ *and illustrated in Figure 5.7(a). According to the aggregation function illustrated in Table 5.2, the overall robustness of such a sequential composition is defined by*

$$\begin{aligned} Q_r(c^1_{Sequence}) &= \min\{q_r(sl^1_{1,2}), q_r(sl^1_{2,3}), q_r(sl^1_{3,5}), q_r(sl^1_{5,6}), q_r(sl^1_{6,8})\} \\ &= \min\{0, 1, 1, 0, 0\} \\ &= 0 \end{aligned} \tag{5.11}$$

By computing the robustness $Q_r(c^1_{AND})$ *(see Table 5.2) of the AND-Branching composition* c^1_{AND}*, extracted from* c^1*, defined by* $\{< T_5, s_5 >, < T_6, s_6 >, < T_7, s_7 >\}$ *and illustrated in Figure 5.7(b) we obtain* 0.

Example 53. ***(Robustness of OR-Branching Composition)*** *Let* c^1_{OR} *be the OR-Branching composition. Such a composition extracted from* c^1 *is defined by* $\{< T_1, s_1 >, < T_2, s_2 >, < T_4, s_4 >\}$ *and illustrated in Figure 5.7(c). Assuming that each*

[8] Our illustrations are performed on different standard compositions described in Figure 5.7. These compositions are extracted from composition illustrated in Figure 5.4.

OR-branch of c^1_{OR} *has the same probability (i.e.,* $p_{sl^k_{i,j}} = \frac{1}{|sl^k_{i,j}|}$ *for each semantic link* $sl^k_{i,j}$ *of* c^1_{OR}*) to be used, the overall robustness of such an OR-Branching composition is defined by (5.12) (see aggregation function illustrated in Table 5.2).*

$$\begin{aligned} Q_r(c^1_{OR}) &= \frac{1}{2} \times (q_r(sl^1_{1,2})) + \frac{1}{2} \times (q_r(sl^1_{1,4})) \\ &= \frac{1}{2} \times (0) \\ &= 0 \end{aligned} \tag{5.12}$$

ii) **Common Description rate.**

- On the one hand the Common Description rate Q_{cd} of both a sequential $c_{Sequence}$ and an AND-Branching composition c_{AND} is defined as the *weighted average* of its semantic links $sl_{i,j}$'s Common Description rate $q_{cd}(sl^k_{i,j})$.

$$\begin{aligned} Q_{cd}(c_{Sequence}) &= Q_{cd}(c_{AND}) \\ &= \frac{1}{|sl^k_{i,j}|} \sum_{sl^k_{i,j}} q_{cd}(sl^k_{i,j}) \end{aligned} \tag{5.13}$$

  In this direction all different rates are considered with same importance, depending on the involvement of the semantic links on the overall composition. Therefore the overall common description rate of the latter compositions $c_{Sequence}$ or c_{AND} is depending on a linear function of semantic links' common description rate. Indeed all different rates involved in the composition $c_{Sequence}$ or c_{AND} require to be considered together i) in the same way and ii) in a linear function. We can sketch the average rate of common description the composition has for each of these semantic links.

- On the other hand the common description rate Q_{cd} of an OR-Branching semantic link composition c_{OR} is a *sum of* $q_{cd}(sl^k_{i,j})$ *weighted by* $p_{sl^k_{i,j}}$.

$$Q_{cd}(c_{OR}) = \sum_{sl^k_{i,j}} q_{cd}(sl^k_{i,j}).p_{sl_{i,j}} \tag{5.14}$$

  Therefore each conditional branch is weighted by $p_{sl^k_{i,j}}$ i.e., the *probability that the semantic link* $sl^k_{i,j}$ *be chosen at run time.*

Example 54. ***(Common Description rate of Sequential and AND-Branching Composition)***
Let $c^1_{Sequence}$ *be the sequential composition illustrated in Figure 5.7(a). According to the aggregation function illustrated in Table 5.2, the overall common description rate of such a sequential composition is defined by*

$$\begin{aligned} Q_{cd}(c^1_{Sequence}) &= \frac{1}{5} \times (q_{cd}(sl^1_{1,2}) + q_{cd}(sl^1_{2,3}) + q_{cd}(sl^1_{3,5}) + q_{cd}(sl^1_{5,6}) + q_{cd}(sl^1_{6,8})) \\ &= \frac{1}{5} \times (\frac{5}{9} + 1 + 1 + \frac{3}{5} + \frac{1}{5}) \\ &= \frac{151}{225} \end{aligned} \tag{5.15}$$

By computing the common description rate $Q_{cd}(c)^1_{AND}$ *(see Table 5.2) of the AND-Branching composition* c^1_{AND} *illustrated in Figure 5.7(b) we obtain* $\frac{31}{70}$.

Example 55. ***(Common Description rate of OR-Branching Composition)*** *Let c^1_{OR} be the OR-Branching composition illustrated in Figure 5.7(c). Assuming that each OR-branch of c^1_{OR} has the same probability, the overall robustness of such an OR-Branching composition is defined by (5.16).*

$$\begin{aligned} Q_{cd}(c^1_{OR}) &= \frac{1}{2} \times (q_{cd}(sl^1_{1,2})) + \frac{1}{2} \times (q_{cd}(sl^1_{1,4})) \\ &= \frac{1}{2} \times (\frac{5}{9}) + \frac{1}{2} \times (\frac{5}{9}) \\ &= \frac{5}{9} \end{aligned} \tag{5.16}$$

iii) **Matching Quality.**

- The matching quality Q_m of a sequential $c_{Sequence}$ and AND-Branching semantic link composition c_{AND} is defined as a *product* of $q_m(sl^k_{i,j})$.

$$\begin{aligned} Q_m(c_{Sequence}) &= Q_m(c_{And}) \\ &= \prod_{sl^k_{i,j}} q_m(sl^k_{i,j}) \end{aligned} \tag{5.17}$$

  Contrary to the robustness and common description rate of sequential and AND-Branching composition, the overall matching quality of the latter composition is depending on a nonlinear function of semantic links' matching quality. All different (non empty) matching qualities involved in the composition $c_{Sequence}$ or c_{AND} require to be considered together in such a non linear aggregation function to make sure that compositions that contains semantic links with low or high matching quality will be more easily identified, and then pruned. Even if robustness and matching quality are closed functional criteria, their aggregation functions are not. Therefore the aggregation functions of robustness and matching quality focus on different properties.
- On the other hand the matching quality Q_m of an OR-Branching semantic link composition c_{OR} is a *sum of* $q_m(sl^k_{i,j})$ *weighted by* $p_{sl^k_{i,j}}$.

$$Q_m(c_{OR}) = \sum_{sl^k_{i,j}} q_m(sl^k_{i,j}).p_{sl_{i,j}} \tag{5.18}$$

  Therefore each conditional branch is weighted by $p_{sl^k_{i,j}}$ i.e., the *probability that the semantic link $sl^k_{i,j}$ be chosen at run time.*

Example 56. ***(Matching Quality of Sequential and AND-Branching Composition)***

Let $c^1_{Sequence}$ be the sequential composition illustrated in Figure 5.7(a). According to the aggregation function illustrated in Table 5.2, the overall matching quality of such a sequential composition is defined by

$$\begin{aligned} Q_m(c^1_{Sequence}) &= q_m(sl^1_{1,2} \times q_m(sl^1_{2,3}) \times q_m(sl^1_{3,5}) \times q_m(sl^1_{5,6}) \times q_m(sl^1_{6,8}) \\ &= \frac{1}{2} \times 1 \times 1 \times \frac{1}{2} \times \frac{1}{4} \\ &= \frac{1}{16} \end{aligned} \tag{5.19}$$

Composition Construct	Quality Criterion		
	Robustness Q_r	Common Descritption rate Q_{cd}	Matching Quality Q_m
Sequential/ AND- Branching	$\min_{sl^k_{i,j}}\{q_r(sl^k_{i,j})\}$	$\frac{1}{\lvert sl^k_{i,j}\rvert}\sum_{sl^k_{i,j}} q_{cd}(sl^k_{i,j})$	$\prod_{sl^k_{i,j}} q_m(sl^k_{i,j})$
OR-Branching	$\sum_{sl^k_{i,j}} q_r(sl^k_{i,j}).p_{sl^k_{i,j}}$	$\sum_{sl^k_{i,j}} q_{cd}(sl^k_{i,j}).p_{sl^k_{i,j}}$	$\sum_{sl^k_{i,j}} q_m(sl^k_{i,j}).p_{sl_{i,j}}$

Table 5.2: Quality Aggregation Rules for Semantic Link Composition.

By computing the overall matching quality $Q_m(c^1_{AND})$ (see Table 5.2) of the AND-Branching composition c^1_{AND} illustrated in Figure 5.7(b) we obtain $\frac{3}{8}$.

Example 57. ***(Matching Quality of OR-Branching Composition)***
Let c^1_{OR} be the OR-Branching composition illustrated in Figure 5.7(c). Assuming that each OR-branch of c^1_{OR} has the same probability, the overall matching quality of such an OR-Branching composition is defined by (5.20).

$$\begin{aligned} Q_m(c^1_{OR}) &= \frac{1}{2} \times (q_m(sl^1_{1,2})) + \frac{1}{2} \times (q_m(sl^1_{1,4})) \\ &= \frac{1}{2} \times (\frac{1}{2}) + \frac{1}{2} \times (\frac{1}{2}) \\ &= \frac{1}{2} \end{aligned} \tag{5.20}$$

Using the above aggregation functions, the quality vector of a practical semantic link composition is defined by (5.21).

$$Q(c) = (Q_r(c), Q_{cd}(c), Q_m(c)) \tag{5.21}$$

All the previous criteria $q_{l,l\in\{r,cd,m\}}$ are said "*positive*" [209] i.e., the higher the value Q_l for c the higher its l^{th} quality.

Although the introduced quality model for semantic link composition has specific function aggregations (i.e., Q_r, Q_{cd}, Q_m which are based on heuristics), it is straightforward i) to adapt these functions depending on the formalization requirements (e.g., other heuristics) and ii) to extend the number of aggregation functions in case one considers more than three quality criteria. For instance the quality Q_{cd} of sequential, AND-Branching and OR-Branching compositions can be computed by valuing the upper bound of common description of their semantic links.

5.2.3 Partial Independence of Quality Criteria and their Aggregation Functions

The three introduced quality criteria used to value Web service composition focus on an unique criterion of semantic composability i.e., the semantic link. Therefore Robustness, Common Description rate and Matching Quality may seem highly dependent but they are not, especially their aggregation functions.

On the one hand we address (not formally) the (relative) independence of these criteria. First of all the independence of i) Robustness and Common Description rate criteria, ii) Matching Quality and Common Description rate criteria are both established on individual semantic links, hence on their composition. On the other hand the independence of iii) Matching Quality and

Robustness is established on their composition. Indeed robustness and matching quality are highly dependent since a semantic link with a matching quality higher than $\frac{3}{4}$ is automatically known as robust.

Robustness and Common Description Rate

The *Robustness* and *Common Description Rate* have no direct relationship between them. Indeed a robust semantic link with very few common descriptions can be retrieved in a composition. In the same composition, a non robust semantic link with a higher rate of common description can be retrieved as well. This ensures their independence.

Common Description Rate and Matching Quality

In the same way as the previous pair of quality criteria, the common description rate has not direct impact on the matching quality and vice versa.

Example 58. ***(Independence of Common Description Rate and Matching Quality)*** *As we remark the quality of matching of two semantic links can be very different (e.g., from single to double for $sl^1_{1,2}$ and $sl^3_{1,2}$), and have a common description rate very close (see (5.2) and (5.3)). In the same way we can also obtain a low level of quality of matching and a high level of common description, and vice versa. Therefore such criteria are obviously independent.*

Matching Quality and Robustness

Even if criteria q_r, q_m used to value a single semantic link have some dependences, their aggregated values of compositions Q_r, Q_m for Sequential, and AND-Branching are independent since their values are computed from different aggregation functions i.e., minimum for Q_r, product for Q_m. Thus a composition c with a high robustness may have either a high or low overall matching quality as well.

In the special case of a composition with only OR-Branching, the matching quality can be inferred from robustness and vice versa. A way to overcome this special case is to consider, for instance, the minimum of robustness values of the OR-Branching composition.

5.3 Semantic Link Selection

In the following we study the optimal composition[9] as the selection of semantic links that optimize the overall quality of the composition. First of all we briefly remind the main concepts and limitations of the local selection based approach. Then we focus on the naive global approach and its exhaustive search. Finally we overcome issues related to local and naive based approaches by presenting a novel integer linear programming (IP) [204, 96] based global selection. This is performed by a selection of semantic links without generating all practical compositions. This approach i) further constrains semantic links, and ii) meets a given objective (a quality threshold). Moreover such an approach avoids some obvious computational problems associated with the naive approach.

[9]The relation of quality of composition with quality of services is not addressed in this thesis, but addressed in Future Work.

5.3.1 Local Selection Based Approach and its Limitations

In order to select the optimal practical composition, we can make the selection by local optimization approach. Therefore the selection is locally optimized at each abstract semantic link $sl^A_{i,j}$ of the composition. Roughly speaking the selection is done on candidate semantic links that have the highest quality vector for each abstract semantic link. The selection process ends with a practical composition wherein all semantic links are the best local links. Nevertheless the local optimization approach has two shortcomings.

First, the local selection of a candidate semantic link $sl^k_{i,j}$ enforces a specific service for both tasks T_i and T_j. Such a selection constrains not only i) the services for T_i and T_j but also ii) the output parameter of semantic link $sl^k_{i,j}$ and iii) the input parameter of $sl^k_{i,j}$. Thus, these constraints can no longer ensure to select neither the best links for its closest abstract semantic links $sl^A_{\alpha,i}$ and $sl^A_{j,\beta}$ nor the optimal semantic links based composition. In other words the practical composition created by local optimization may not be the best *semantic* composition we can choose. The quality value of the composition which is formed by semantic links, may not be the global optimum composition.

Example 59. ***(A Drawback of the Local Selection Based Approach)*** *Suppose c^A be the sequential composition of T_1, T_2 and T_3 illustrated in Figure 5.8 (i.e., a part of the abstract and sequential composition illustrated in Figure 5.7(a)). The best local selection for abstract semantic link $sl^A_{1,2}$ is $sl^2_{1,2}$ for any quality criterion. The resulting composition c^1 consists of $sl^2_{1,2}$ and $sl^2_{2,3}$. This composition is valued by a quality vector $(0, \frac{4}{5}, \frac{1}{4})$. A concurrent practical composition c^2 which consists of $sl^1_{1,2}$ and $sl^1_{2,3}$ is valued by a quality vector $(0, \frac{7}{9}, \frac{1}{2})$. However c^1 does not lead to the optimal composition for each criterion. Indeed the overall matching quality value of the practical composition c^1 is lower than the practical composition c^2. In such cases the local selection based approach is not appropriate.*

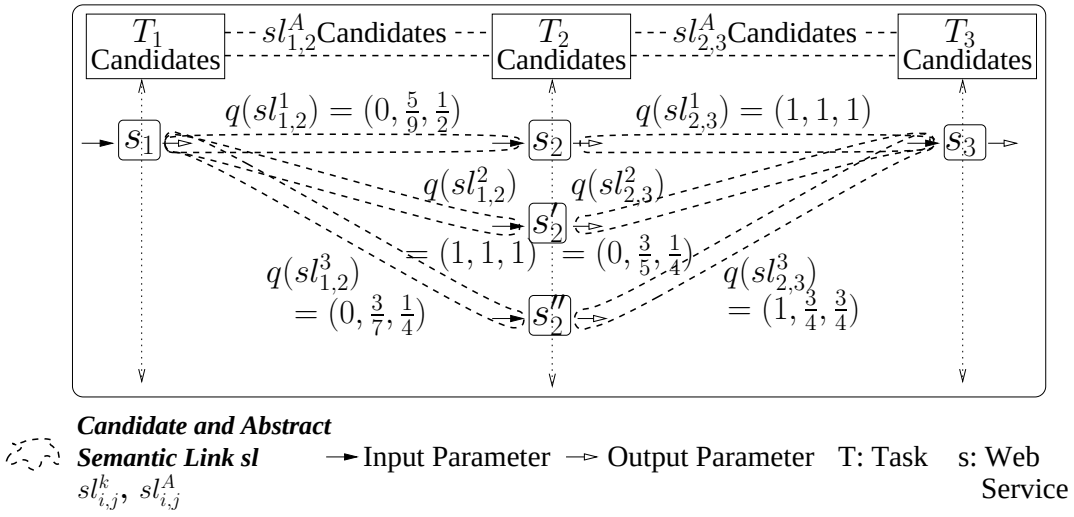


Figure 5.8: Tasks, Candidate Web Services and Semantic Links.

Secondly, during the selection of semantic links, the local optimization approach can consider constraints on individual abstract semantic links, but it cannot consider global constraints, i.e.,

constraints that cover multiple (or all) abstract semantic links in the composite service. Also, although it is always able to select a semantic link with maximal common description rate or maximal matching quality for each abstract semantic link, it fails when both the common description rate and maximal matching quality need to be considered at a global level. The global quality constraints may be then not satisfied, leading to a suboptimal composition (could lead to a global constraint violation). For instance, a global constraint with a common description rate higher than 70% and a matching quality higher than 10% cannot be enforced.

In the rest of this section we study global based approaches that overcome the previous shortcoming to select practical compositions.

5.3.2 Naive Global Selection Based Approach

For each abstract semantic link $sl^A_{i,j}$ in an abstract composition, there is a set of candidate semantic links $sl^{k,1\leq k\leq n}_{i,j}$ that can instantiate abstract semantic link $sl^A_{i,j}$. Assigning a candidate semantic link $sl^k_{i,j}$ to each abstract semantic link $sl^A_{i,j}$ in an abstract composition leads to a possible practical composition. In the (naive) global selection based approach, all possible practical compositions are generated (at least conceptually speaking) and the one which maximizes the user's preferences while satisfying the imposed constraints is then selected.

Example 60. ***(A Set of Possible Practical Compositions)*** *Assume an abstract composition c^A illustrated in Figure 5.2. Moreover we assume all candidate semantic links defined in Table 5.1. According to this Table we can obtain three practical compositions c^1, c^2 and c^3 for c^A. Each of these compositions requires the same semantic link for abstract semantic links $sl^A_{1,4}$, $sl^A_{3,5}$, $sl^A_{4,5}$, $sl^A_{5,6}$, $sl^A_{5,7}$, $sl^A_{6,8}$, $sl^A_{7,8}$ i.e., $sl^1_{1,4}$, $sl^1_{3,5}$, $sl^1_{4,5}$, $sl^1_{5,6}$, $sl^1_{5,7}$, $sl^1_{6,8}$, $sl^1_{7,8}$. On the contrary the semantic links required by $sl^A_{1,2}$ and $sl^A_{2,3}$ differs for c^1, c^2 and c^3. Indeed c^1, c^2 and c^3 require respectively:*

- $sl^1_{1,2}$ *for* $sl^A_{1,2}$ *and* $sl^1_{2,3}$ *for* $sl^A_{2,3}$*;*
- $sl^2_{1,2}$ *for* $sl^A_{1,2}$ *and* $sl^2_{2,3}$ *for* $sl^A_{2,3}$*;*
- $sl^3_{1,2}$ *for* $sl^A_{1,2}$ *and* $sl^3_{2,3}$ *for* $sl^A_{2,3}$.

Even if $sl^A_{1,2}$ and $sl^A_{2,3}$ can be both instantiated by three semantic links, we do not obtain nine potential practical compositions of Web services. Indeed, defining a service for a given task (e.g., s_2 for T_2) directly constrains the incoming and outgoing semantic links (e.g., $sl^1_{1,2}$ for $sl^A_{1,2}$ and $sl^1_{2,3}$ for $sl^A_{2,3}$).

In the naive approach the selection of a practical composition relies on the application of a Multiple Criteria Decision Making technique (MCDM) [85] on a quality matrix

$$\mathbf{Q} = (Q^\lambda_l;\ 1 \leq \lambda \leq p,\ l \in \{r, cd, m\}) \tag{5.22}$$

of the abstract composition. In this matrix, a row λ corresponds to the quality vector of a possible practical composition (actually the λ^{th} practical compositions among the p potential compositions) for the abstract composition. The previous vector has been computed by merging the quality vectors of some candidate semantic links. This has been achieved by means of aggregation functions illustrated in Table 5.2. Each column $l \in \{r, cd, m\}$ of the matrix corresponds to a quality dimension. They respectively refer to *Robustness*, *Common Description rate* and *Matching Quality*.

Example 61. ***(Quality Matrix of a Set of Possible Practical Compositions)***
The quality matrix $\boldsymbol{Q}$ *of compositions* c^1, c^2 *and* c^3 *illustrated in Example 60 is defined as follows:*

$$\boldsymbol{Q} = \begin{pmatrix} Q_r^1 & Q_{cd}^1 & Q_m^1 \\ Q_r^2 & Q_{cd}^2 & Q_m^2 \\ Q_r^3 & Q_{cd}^3 & Q_m^3 \end{pmatrix} = \begin{pmatrix} Q_r(c^1) & Q_{cd}(c^1) & Q_m(c^1) \\ Q_r(c^2) & Q_{cd}(c^2) & Q_m(c^2) \\ Q_r(c^3) & Q_{cd}(c^3) & Q_m(c^3) \end{pmatrix} \tag{5.23}$$

i.e.,

$$\boldsymbol{Q} = \begin{pmatrix} 0 & 0.6189 & 0.0087 \\ 0 & 0.6227 & 0.0058 \\ 0 & 0.5875 & 0.0051 \end{pmatrix} \tag{5.24}$$

From this matrix , the naive process of selection requires a step of *Simple Additive Weighting* (SAW) [85] to select an optimal practical composition. The two phases required by SAW are:

1. **Scaling Phase**. We first scale the values of each quality criterion. Since the suggested criteria are positive (see Section 5.2.2), quality values $Q_r^\lambda, Q_{cd}^\lambda, Q_m^\lambda$ are scaled according to (5.25).

$$\tilde{Q}_l^{\lambda} = \begin{cases} \frac{Q_l^\lambda - Q_l^{\min}}{Q_l^{\max} - Q_l^{min}} & \text{if } Q_l^{\max} - Q_l^{min} \neq 0 \\ 1 & \text{if } Q_l^{\max} - Q_l^{min} = 0 \end{cases} \quad l \in \{r, cd, m\} \tag{5.25}$$

   In (5.25), $Q_l^{\max}$ is the maximal value of the l^{th} quality criteria whereas $Q_l^{\min}$ is the minimal value of the l^{th} quality criteria. This scaling phase complexity is linear in the number of abstract semantic links in the composite service. By applying this equation on $\mathbf{Q}$, we obtain a matrix

$$\tilde{Q} = (\tilde{Q}_l^{\lambda}; 1 \leq \lambda \leq p, l \in \{r, cd, m\}) \tag{5.26}$$

   in which each row $\tilde{Q}^{\lambda}$ corresponds to a practical composition c^λ while each column $\tilde{Q}_l$ corresponds to a quality dimension.

   Example 62. ***(Scaling Phase Illustration)***
   Here we apply equation 5.25 to quality matrix $\boldsymbol{Q}$ *previously computed in Example 61. Therefore we obtain* $\overset{\sim}{\boldsymbol{Q}}$ *in equations* (5.27) *and* (5.28).

$$\overset{\sim}{\boldsymbol{Q}} = \begin{pmatrix} \tilde{Q}_r^1 & \tilde{Q}_{cd}^1 & \tilde{Q}_m^1 \\ \tilde{Q}_r^2 & \tilde{Q}_{cd}^2 & \tilde{Q}_m^2 \\ \tilde{Q}_r^3 & \tilde{Q}_{cd}^3 & \tilde{Q}_m^3 \end{pmatrix} \tag{5.27}$$

   i.e.,

$$\overset{\sim}{\boldsymbol{Q}} = \begin{pmatrix} 1 & 0.8920 & 1 \\ 1 & 1 & 0.1944 \\ 1 & 0 & 0 \end{pmatrix} \tag{5.28}$$

   Note that we can compute the value of $Q_l^{\max}$ and $Q_l^{\min}$ in these equations without generating all possible practical compositions. For example, in order to compute the maximum common description rate (i.e., $Q_{cd}^{\max}$) of all the practical compositions, we select the semantic link with the highest rate of common description for each abstract semantic link $sl_{i,j}^A$ and sum up all these rates to compute $Q_{cd}^{\max}$. The computation cost of $Q_l^{\max}$ and $Q_l^{\min}$ is thus polynomial.

2. **Weighting Phase**. The following formula is used to compute the overall quality score for each practical composition c^{λ}:

$$Score(c^{\lambda}) = \sum_{l \in \{r,cd,m\}} \left(\widetilde{Q}_l^{\lambda} \times \omega_l \right) \tag{5.29}$$

$\omega_l \in [0,1]$ is the weight assigned to the l^{th} quality criterion and $\sum_{l \in \{r,cd,m\}} \omega_l = 1$. In this way preferences on quality of the desired practical compositions (i.e., balance the impact of the different criteria) can be done by simply adjusting ω_l e.g., the Common Description rate could be weighted higher.

The global naive approach will choose the practical composition which has the maximal value of $Score(c^{\lambda})$ (i.e., $\max(Score(c^{\lambda}))$). If there is more than one practical composition which has the same maximal value of $Score(c^{\lambda})$, then a practical composition will be selected from them randomly.

5.3.3 Integer Programming Based Global Selection

The naive global selection approach by exhaustive searching outlined above requires the generation of all possible practical compositions. Let $|sl_{i,j}^{A}|$ be the number of abstract semantic links in a composition and n be the minimal number of candidate services by task, the total number of practical semantic link compositions is at least $n^{2.|sl_{i,j}^{A}|}$, making this approach impractical for large scale compositions.

Accordingly, we propose a method based on Integer Programming for selecting an optimal practical composition without generating all possible compositions. The main idea is as follows:

> *The suggested problem requires an objective function maximizing the overall quality subject to semantic links constraints.*

In the following we suggest to adopt global optimization to select the practical composition to overcome the latter issues.

There are three inputs in an IP (Integer linear Programming) problem: an *objective function*, a set of integer decision *variables* (restricted to value 0 or 1), and a set of *constraints* (equalities or inequalities), where both the objective function and the constraints must be linear. IP attempts to maximize or minimize the value of the objective function by adjusting the values of the variables while enforcing the constraints. The outputs of an IP problem are as follows:

- the maximum (or minimum) value of the objective function;
- the values of variables at this maximum (minimum).

The problem of selecting an optimal practical composition is mapped into an IP problem.

Objective Function

Here we suggest to first formalize the objective function of the IP problem. In the same way as the naive approach, the robustness, common description rate and matching quality values of the p potential practical compositions i.e., $Q_{l,l \in \{r,cd,m\}}^{\lambda,1 \leq \lambda \leq p}$ have been first determined by means of aggregation functions in Table 5.2. We also rely on MCDM and SAW techniques to determine

the desirability of a practical composition. In addition we use the function (5.30) to formalize the objective function of the IP problem. This is based on (5.25) and (5.29).

$$\max_{1\leq\lambda\leq p}\left(\sum_{l\in\{r,cd,m\}}\left(\widetilde{Q}_l^{\lambda}\times\omega_l\right)\right) \tag{5.30}$$

where $\omega_l \in [0,1]$ and $\sum_{l\in\{r,cd,m\}}\omega_l = 1$. ω_l is the weight assigned to the l^{th} quality criterion.

The objective function will be the score of the best practical composition of Web services i.e., the best weighted sum of the scaled values of quality dimension.

Integer Variables & Constraints of IP Problem

The remainder of this subsection describes the integer variables and constraints of the IP problem.

For every candidate semantic link $sl_{i,j}^{k,1\leq k\leq n}$ of an abstract semantic link $sl_{i,j}^{A}$, we include an integer variable $y_{i,j}^{k}$ in the IP problem indicating the selection or exclusion of semantic link $sl_{i,j}^{k}$. By convention $y_{i,j}^{k}$ is 1 if the k^{th} candidate semantic link $sl_{i,j}^{k}$ is selected to concretize $sl_{i,j}^{A}$ between tasks T_i and T_j, 0 otherwise. The semantic links which are selected to achieve abstract semantic links of the abstract composition are the outputs of the IP problem. They will form an optimal practical composition satisfying (5.30) and meeting the following constraints: *Allocation Constraint*, *Incompatibility Constraint*, *Robustness Constraint*, *Common Description Rate Constraint*, *Matching Quality Constraint*, *Local Constraint*.

- **Allocation Constraint.**

  For each abstract semantic link $sl_{i,j}^{A}$ between two tasks T_i and T_j, there is a set of potential semantic $sl_{i,j}^{k}$ that can be assigned (allocated) depending on selected Web services in T_i and T_j. However only one candidate semantic link should be selected for each abstract semantic link $sl_{i,j}^{A}$ between tasks T_i and T_j. Given that the integer variables $y_{i,j}^{k,1\leq k\leq n}$ denotes the selection of semantic link $sl_{i,j}^{k,1\leq k\leq n}$ for abstract semantic link $sl_{i,j}^{A}$, the constraint formalized in (5.31) must be satisfied:

  $$\sum_{k=1}^{n} y_{i,j}^{k} = 1,\ \forall sl_{i,j}^{A} \tag{5.31}$$

  where n is the number of candidate semantic links for $sl_{i,j}^{A}$.

  Example 63. ***(Allocation Constraint)***
  Suppose the abstract sequential composition of tasks T_1, T_2 and T_3 illustrated in Figure 5.8. Three candidate semantic links can be applied between tasks T_1 and T_2 i.e., $sl_{1,2}^{1}$, $sl_{1,2}^{2}$ and $sl_{1,2}^{3}$. Since only one candidate semantic link between two tasks will be selected, we have the following allocation constraint:

  $$y_{1,2}^{1} + y_{1,2}^{2} + y_{1,2}^{3} = 1 \tag{5.32}$$

  In the same way we obtain the allocation constraint

  $$y_{2,3}^{1} + y_{2,3}^{2} + y_{2,3}^{3} = 1 \tag{5.33}$$

  for abstract semantic link $sl_{2,3}^{A}$.

- **Incompatibility Constraint.**

  Since the selection of a candidate semantic link $sl^k_{i,j}$ for $sl^A_{i,j}$ enforces a specific Web service for both tasks T_i (e.g., s_i) and T_j (e.g., s_j), the number of candidate semantic links concretizing the closest abstract semantic links of $sl^A_{i,j}$ (i.e., $sl^A_{\alpha,i}$ and $sl^A_{j,\beta}$) is highly reduced. Indeed the candidate semantic links for $sl^A_{j,\beta}$ have to use only one of the output parameters of s_j. Therefore the semantic link used by abstract semantic link $sl^A_{j,\beta}$ is defined by $sl^{k_2}_{j,\beta}$ i.e., $\langle s_j, Sim_{\mathcal{T}}(Out\_s_j, In\_s_\beta), s_\beta\rangle^{k_2}$. In the same way the candidate semantic links for $sl^A_{\alpha,i}$ have to use only one of the input parameter of s_i. Therefore the semantic link used by abstract semantic link $sl^A_{\alpha,i}$ is defined by $sl^{k_1}_{\alpha,i}$ i.e., $\langle s_\alpha, Sim_{\mathcal{T}}(Out\_s_\alpha, In\_s_i), s_i\rangle^{k_1}$. Figure 5.9 illustrates this constraint.

  A constraint (5.34) for each pair of incompatible candidate semantic links $(sl^{k_1}_{i,j}, sl^{k_2}_{j,\beta})$ is required in our IP problem. Such a constraint is formalized as follows:

$$y^{k_1}_{i,j} + y^{k_2}_{j,\beta} \leq 1,\ \forall sl^A_{i,j}\ \forall sl^A_{j,\beta} \tag{5.34}$$

Example 64. ***(Incompatibility Constraint)***
Suppose the composition in Figure 5.8. According to (5.34), the six incompatibility constraints are:

i) $y^1_{1,2} + y^2_{2,3} \leq 1$ *iii)* $y^2_{1,2} + y^1_{2,3} \leq 1$ *v)* $y^3_{1,2} + y^1_{2,3} \leq 1$
ii) $y^1_{1,2} + y^3_{2,3} \leq 1$ *iv)* $y^2_{1,2} + y^3_{2,3} \leq 1$ *vi)* $y^3_{1,2} + y^2_{2,3} \leq 1$

Indeed $(sl^1_{1,2}, sl^2_{2,3})$ *is a pair of incompatible candidate semantic links since i) the selection of one semantic link for* $sl^A_{1,2}$ *reduces the set of possible service for* T_2*, hence the set of possible semantic links for* $sl^A_{2,3}$ *and ii) task* T_2 *cannot be performed by two distinct services* s_a *and* s_b*.*

In other words either $sl^1_{1,2}$ *or* $sl^2_{2,3}$ *is selected, but not both i.e., either* $y^1_{1,2}$ *or* $y^2_{2,3}$ *is equal to one but not both.*

We have the same explanation for the five other constraints.

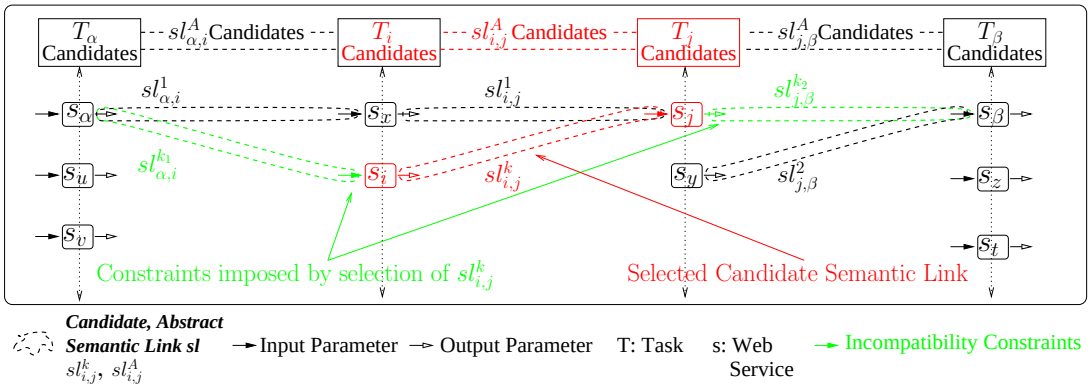


Figure 5.9: Illustration of an Incompatibility Constraint.

Besides (5.31) and (5.34), IP constraints on each quality criterion of the whole abstract composition are required. Here, we focus on the sequential and AND-Branching compositions,

but a similar (straightforward) formalization for OR-Branching compositions and a fortiori their combinations (depending on the target problem) is required.

- **Robustness Constraint.**

  Let $r^k_{i,j}$ be a function of (i,j,k) representing the robustness quality of a semantic link $sl^k_{i,j}$. Constraint (5.35) is required to capture the robustness quality of a semantic link composition.

$$Q_r = \frac{1}{|sl^A_{i,j}|} \sum_{sl^A_{i,j}} \sum_{k=1}^{n} r^k_{i,j}.y^k_{i,j} \tag{5.35}$$

  An additional constraint of (5.35) i.e., (5.36) can be used to constrain the practical composition to be robust. This *hard* constraint ensures to compute robust composition of Web services.

$$\frac{1}{|sl^A_{i,j}|} \sum_{sl^A_{i,j}} \sum_{k=1}^{n} r^k_{i,j}.y^k_{i,j} = 1 \tag{5.36}$$

- **Common Description Rate Constraint.**

  Let $cd^k_{i,j}$ be a function of (i,j,k) representing the Common Description rate of a link $sl^k_{i,j}$. Its constraint is defined in the same way as (5.35) and (5.36) by replacing Q_r by Q_{cd}, $r^k_{i,j}$ by $cd^k_{i,j}$. Therefore:

$$Q_{cd} = \frac{1}{|sl^A_{i,j}|} \sum_{sl^A_{i,j}} \sum_{k=1}^{n} cd^k_{i,j}.y^k_{i,j} \tag{5.37}$$

  An additional constraint of (5.37) i.e., (5.38) can be used to constrain the common description rate of the practical composition to not be lower than L.

$$\frac{1}{|sl^A_{i,j}|} \sum_{sl^A_{i,j}} \sum_{k=1}^{n} cd^k_{i,j}.y^k_{i,j} \geq L,\ L \in [0,1] \tag{5.38}$$

- **Matching Quality Constraint.**

  Among the criteria used to select semantic links, the Matching quality is associated with a nonlinear aggregation function (see Table 5.2). A transformation in a linear function is then required to capture it in the IP problem. Assume $m^k_{i,j}$ be a function of (i,j,k) representing the Matching quality of semantic link $sl^k_{i,j}$. The overall Matching quality of the practical composition is formalized by (5.39).

$$Q_m = \prod_{sl^A_{i,j}} \Big(\prod_{k=1}^{n} (m^k_{i,j})^{y^k_{i,j}} \Big) \tag{5.39}$$

The Matching quality constraints can be linearised by applying the logarithm function ln. The equation (5.39) then becomes:

$$\ln(Q_m) = \sum_{sl^A_{i,j}} \left(\sum_{k=1}^{n} \ln(m^k_{i,j}).y^k_{i,j} \right) \tag{5.40}$$

since $\sum_{k=1}^{n} y^k_{i,j} = 1$ and $y^k_{i,j} = 1$ or 0 for each semantic link $sl^k_{i,j}$. $\ln(Q_m)$ is formalized to capture the Matching quality in our work.

Changing a nonlinear constraint in its linear form requires also to update and to linearise the objective function. Therefore, equation (5.41) is replaced by equation (5.42) in (5.25).

$$\frac{Q^\lambda_m - Q^{\min}_m}{Q^{max}_m - Q^{\min}_m} \tag{5.41}$$

$$\frac{\ln(Q^\lambda_m) - \ln(Q^{\min}_m)}{\ln(Q^{max}_m) - \ln(Q^{\min}_m)} \tag{5.42}$$

- **Local Constraint.**

  The IP problem can also include local selection and encompass local constraints. Such constraints can then predicate on properties of a single semantic link and can be formally included in the model. In case a target semantic link $sl^A_{i,j}$ requires its local common description rate to be higher than a given value v, this constraint is defined by (5.43).

$$\sum_{k=1}^{n} cd^k_{i,j}.y^k_{i,j} > v, \ v \in [0,1] \tag{5.43}$$

  Local constraints such as (5.43) can be applied on any quality criterion. Such constraints are enforced during the semantic links selection. Those which violate the local constraints are filtered from the list of candidate semantic links, reducing the number of variables of the model.

The proposed method for translating the problem of selecting an optimal execution composition into an IP problem is generic and, although it has been illustrated with criteria introduced in Section 5.2, other criteria of semantic quality to value semantic links can be accommodated.

Computational Complexity

The computational cost of local optimization is polynomial whereas the optimization problem formulated in Section 5.3.3, which is equivalent to Integer Linear Programming and multiple knapsack problems [196], is NP-hard [43, 155]. In case the number of abstract and candidate semantic links in the system is expected to be very high (the order of millions or higher), finding the exact optimal solution to the optimization problem takes exponential run-time complexity in the worst case, and is not practical. Instead our approach scales well by running a heuristic based IP solver wherein hundreds of abstract and candidate semantic links are involved (see the Experiments Chapter 7 and the Section 7.3.3). This is a suitable upper bound for practicable industrial applications. The computational performance of our approach can be further improved, for instance, by dividing the problem into several *global selection problems*.

5.4 Conclusion

5.4.1 Synthesis

Limitation and Future Work

Since several practical compositions maximizing the overall quality of semantic links may be computed, the main direction for future work is to consider also optimality for quality of service to further optimize them. The latter notion of optimality for quality of service is, at least in part, driven by empirical analysis of usage of compositions rather than by comparing input-output specifications.

Another issue with global planning is that users are required to provide relatively complex inputs (i.e., global constraints and trade offs).

Finally it would be interesting to focus on a process that reduces the number of services (ideally to one) in any composition, then reducing the number of semantic links and enhancing the semantic link based composition. This can be performed by discovering more general services that achieve a set of (or cluster of) tasks in the abstract composition.

Concluding Remarks

This Chapter has been directed to meet a main challenge facing semantic links based web service composition i.e., how effectively compute optimal compositions of semantic links. Starting from an initial set of web services, the suggested approach aims at selecting web services and maximizing the overall quality of their inter-connections by means of their semantic links according to a goal to achieve. The **requirement** $R_{Optimization}$ is ensured by this approach.

To this end we have first presented a general and extensible model to evaluate quality of both elementary and composition of semantic links. The latter model considers *Robustness*, *Common Description Rate* and *Matching Quality* as main attributes to such an evaluation of *semantics*. Therefore, contrary to most of approaches which focus on non functional criteria such as the quality of service (QoS), we consider the quality of semantic links as an innovative and distinguishing functional criterion to estimate the overall semantic quality of web service compositions. The latter quality is measured along two perspectives:

- **Quality Semantic Criteria**. As we discussed in earlier section, semantic quality of composite service is measured by a set of quality criteria. And since there are often trade offs among different quality criterion (e.g., Common Description Rate and Matching Quality), it is important that the system is able to find a combination of these dimensions that fits the user's preferences.
- **Ability to satisfy user's requirements**. Although good semantic quality composite service requires optimal semantic quality of links, the satisfaction of end users' constraints is an equally important aspect. There are two kinds of constraints: constraints on a single abstract semantic link and constraints on multiple abstract semantic links. The system's ability to accept both kinds of constraints is key to satisfying user requirements.

Then the limitations and the inappropriate status of the local based approach has been presented. Therefore we determine the best concretization of a composite service as a problem of

global semantic link selection. First we focused on a naive global approach as well as its limitations due to an expensive algorithm. Towards such issues the latter problem is formalized as an optimization problem aiming to i) maximize an objective function of the available semantic links attributes (i.e., functional quality criteria) and ii) meet the constraints specified for some of the attributes. In particular, constraints of allocation and incompatibility between semantic links are required in such a problem. Moreover there are the global constraints, i.e. assertions on the overall *semantic quality* attribute values. Local constraints, i.e. constraints on each abstract semantic link composing the composite service, need to be checked when choosing the set of candidate semantic link to bind. Towards this issue Integer Programming techniques are used to compute optimal practical composition of services. However other global optimization techniques [94] such as genetic algorithms, evolutionary algorithms or other heuristics based approaches can be applied to achieve our problem.

Our global selection based approach is not only i) more suitable than approaches in which the semantic links are selected individually and locally in the composite service but also ii) outperforms the naive global approach. As we will see in Section 7.3.3 of Chapter 7 the experimental results show an acceptable computation cost of the IP-based global selection for a high number of abstract and candidate semantic links.

Table 5.3 describes in details the requirements supported by the model introduced in Chapter 5.

5.4.2 Our Contribution in a Nutshell

In this part (Part II), we addressed Web service composition according to six meta requirements: $R^{Composition}_{Automation}$, $R_{Expressivity}$, $R_{Composability}$, $R_{Flexibility}$, $R_{Optimization}$, $R_{Applicability}$.

Towards the issue of automated composition ($R^{Composition}_{Automation}$), we presented two complementary AI planning based composition approaches that deals with different levels of Web service description ($R_{Expressivity^{Service}}$) and different composability criteria ($R_{Composability}$). In particular we further studied the requirement $R_{Composability}$ along two composability criteria i.e., *semantic links* ($R^{Semantic}_{Composability}$) and *causal laws* ($R^{Causal}_{Composability}$) to achieve automation of Web service composition.

The set of computed compositions, also known as composite Web services are defined with expressive control constructs such as sequence, non determinism and concurrency constructs ($R^{Composition}_{Expressivity}$). The requirement $R^{Composition}_{Applicability}$ is studied in Chapter 7.

Finally a process of optimization ($R_{Optimization}$) is suggested to select optimal composition in the latter set of composite Web services.

The composition and optimization approaches satisfy requirement $R_{Flexibility}$ by focusing on upgradeable models. The requirement $R^{Service}_{Applicability}$ is supported by interfacing our approach by standard proposals of Section 1.3.2.

Table 5.4 describes in details the requirements supported by our overall approach introduced in the contribution part (i.e., Chapters 3, 4 and 5 of Part II).

<table>
<tr><th rowspan="2">Requirement R_i</th><th rowspan="2">Details</th><th colspan="2">Chapter 5</th></tr>
<tr><th>Section 5.2</th><th>Section 5.3</th></tr>
<tr><td rowspan="2">$R^{Composition}_{Automation}$</td><td>Formalism</td><td colspan="2" rowspan="2">✗</td></tr>
<tr><td>Composition Mechanism</td></tr>
<tr><td rowspan="2">$R_{Expressivity}$</td><td>$R^{Service}_{Expressivity}$</td><td colspan="2">In this Chapter, input and output parameters are required (with semantic annotation). However preconditions and effects can be further considered to ensure possible and valid compositions of Web services. Information-providing and World-altering Web services.</td></tr>
<tr><td>$R^{Composition}_{Expressivity}$</td><td>✗</td><td>The supported control constructs are as follows: sequence, non determinism, conditional and concurrent compositions.</td></tr>
<tr><td rowspan="2">$R_{Composability}$</td><td>$R^{Semantic}_{Composability}$</td><td colspan="2">Semantic links ar required in this approach. In particular, their valuation is based on matchmaking functions i.e., Exact, PlugIn, Subsume, Disjoint, Intersection, Abduction and Difference. However this set can be further extended. Computation at Design Time</td></tr>
<tr><td>$R^{Causal}_{Composability}$</td><td colspan="2">Causal laws are supported by the approach.</td></tr>
<tr><td colspan="2">$R_{Flexibility}$</td><td colspan="2">Supported by the (flexible and extendable) semantic quality model for elementary semantic links and their composition (heuristics-based).</td></tr>
<tr><td colspan="2">$R_{Optimization}$</td><td>Semantic links based. More specially their Robustness, Common Description rate, and Matching Quality</td><td>Local and global based optimization techniques. Integer Programming techniques are used to compute optimal practical composition of services. However other global optimization techniques [94] can be applied</td></tr>
<tr><td rowspan="2">$R_{Applicability}$</td><td>$R^{Service}_{Applicability}$</td><td colspan="2">Applicable to the OWL-S service profile, WSMO service capability, SWSO Inputs/Outputs or SA-WSDL (see Section 1.3.2 for further details).</td></tr>
<tr><td>$R^{Composition}_{Applicability}$</td><td colspan="2">✗</td></tr>
</table>

Table 5.3: Table of Requirements supported by Chapter 5. Legend: ✗ = not addressed.

<table>
<tr><th rowspan="2">Requirement R_i</th><th rowspan="2">Details</th><th colspan="3">Part II</th></tr>
<tr><th>Chapter 3</th><th>Chapter 4</th><th>Chapter 5</th></tr>
<tr><td rowspan="2">$R^{Composition}_{Automation}$</td><td>Formalism</td><td rowspan="2">✗</td><td>Independent or Situation Calculus</td><td rowspan="2">✗</td></tr>
<tr><td>Composition Mechanism</td><td>(Robust) Ra_4C or s_{sl}Golog + Backward Chaining</td></tr>
<tr><td rowspan="3">$R_{Expressivity}$</td><td>$R^{Service}_{Expressivity}$</td><td>Semantic Annotated Input and Output parameters. Information-providing services.</td><td colspan="2">Semantic Annotated Input, Output parameters, preconditions and effects. Information-providing and World altering Web services.</td></tr>
<tr><td rowspan="2">$R^{Composition}_{Expressivity}$</td><td colspan="2" rowspan="2">The SLM model supports sequence composability of services, non determinism choice of of Web services and concurrent compositions of Web services.</td><td rowspan="2">Sequential, conditional non determinism and concurrent compositions of Web services.</td></tr>
<tr><td>Sequential and conditional compositions.</td></tr>
<tr><td rowspan="2">$R_{Composability}$</td><td>$R^{Semantic}_{Composability}$</td><td colspan="3">Semantic dependence through (valid and robust) semantic links. Valuation with basic matchmaking functions i.e., Exact, PlugIn, Subsume, Disjoint and extra functions i.e., Intersection, Abduction and Difference to compute robust semantic links. Computation at Design Time.</td></tr>
<tr><td>$R^{Causal}_{Composability}$</td><td>✗</td><td colspan="2">Supported by compositions computed with s_{sl}Golog. See Section 4.2 for further details.</td></tr>
<tr><td colspan="2">$R_{Flexibility}$</td><td>Supported by the SLM model.</td><td>Supported by the semantic link and and causal law axioms.</td><td>Supported by the semantic (link) quality model.</td></tr>
<tr><td colspan="2">$R_{Optimization}$</td><td colspan="2">✗</td><td>Semantic links based.</td></tr>
<tr><td rowspan="3">$R_{Applicability}$</td><td rowspan="2">$R^{Service}_{Applicability}$</td><td colspan="3">Applicable to the OWL-S service profile, WSMO service capability, SWSO Inputs/Outputs (see Section 1.3.2 for further details).</td></tr>
<tr><td>SA-WSDL specification.</td><td>SA-WSDL is depending on the approach.</td><td>SA-WSDL specification.</td></tr>
<tr><td>$R^{Composition}_{Applicability}$</td><td colspan="3">BPEL4WS through the BPEL Rendering Component. (See Table 7.1 and Section 7.1.9 of Chapter 7.)</td></tr>
</table>

Table 5.4: Table of Requirements supported by the Contribution Part (i.e., Part II). Legend: ✗ = not addressed.

Part III

Our Approach in Use

Chapter 6

Industrial Scenarios in Use

In this Chapter we present three different scenarios wherein our approach of semantic Web service composition described in the previous chapters (Part II) has been integrated.

These three scenarios respectively refer to:

- a *Telecommunication* scenario which has been **conceived as an *hot*, *reference* and *best* practice (for service-oriented applications) by business units of France Telecom R&D** (especially due to some key requirements). This first scenario **is running in the information system of France Telecom**.
- an *E-Tourism* scenario. This scenario has been studied and implemented in the European project and Network of Excellence *Knowledge Web*[1].
- an *E-HealthCare* scenario, which is has been conceived with some R&D engineers in France Telecom.

In our Ph.D work, we focused on these scenarios since they refer to different application domains (showing the adaptation of our approach to multiple domains) wherein i) the number of semantic Web services is large, ii) the domain ontologies can be easily discovered or built, iii) the computation of robust or valid composition at hand is difficult, and iv) the *return on investment* can be quite fast.

Our system SME^3-Pro[2] (which consists in naive Discovery, Composition and Execution) presented along this Ph.D report is implemented and interacts with Web services dedicated to these different scenarios. Details about the experiments evaluation of our approach on these three different scenarios are presented in Chapter 7.

In the remainder of this Chapter we present in more details i) an introduction of each scenario, ii) the suggested approach , iii) their motivation and description, and iv) their open issues and challenges.

[1] http://knowledgeweb.semanticweb.org/

[2] cf. the SME^3-Pro (**S**e**M**antic w**E**b s**E**rvic**E** **PRO**ject) Open Source Project licensed under the GPL license available at https://sws-orangelabs.elibel.tm.fr/ that is the general framework in which we intend to release the various prototypes produced by our research in area of Semantic Web services.

Note that we focus on the *Telecommunication* scenario since i) it has been illustrated with our composition approach along this Ph.D report (Web services together with its domain ontology have been first presented in Chapter 1), and ii) it is a **real scenario in use in the information system of France Telecom**.

6.1 A Telecommunication Application: Internet Packages

The approach presented in this work is being exploited in the VoIE (Voice over Ip Expertise) Project, conceived by France Telecom R&D.

In the domain of Internet packages, commercial offers proposed by Telecommunication operators are used to be composed of a set of more technical, optional offers. From now Telecommunication operators such as France Telecom suggest only a few restricted set of pre-existing commercial offers. These offers are well known as *All in one* or again *Global Internet Access*[3]. The end users can then choose a pre-existing commercial offer among the available offers. Even if designing pre-defined packaged offers have at least one significant benefit for Telecommunication operators i.e., ease to maintain, this approach is far from convenient for end users. Indeed this set of pre-existing commercial offers does not still satisfy the end user constraints and preferences. Moreover such a method is far from being convenient to support dynamic generation of Internet packages.

6.1.1 Motivation

The main motivation of our work is to give to the end user the possibility to create their own commercial offers according to their real needs, without any kind of assistance. Such an issue is really challenging in the domain of Telecommunication since it addresses other industrial issues related to dynamic and preference-based packaging. Moreover our proposal aims at improving the return on investment (ROI) of any Telecommunication operator by reducing the Time-to-product (dynamic and automated process), Time-to-market, with lower price (repetitive tasks), better quality, better precision, better end-user satisfaction (no constraint-based packages), the whole with more creativity.

In the case under consideration the result of a dynamic and automated generation of commercial offers is a complete customized offer wherein the end user is only in charge of selecting the offer(s) she wants to subscribe. For instance the (non exhaustive set of) offers may be as follows:

- ADSL eligibility (ELIG);
- LiveBox;
- Voice over IP (VOIP);
- Address Book (AB);
- Visiophone (VP);
- Television over IP (TVIP);
- Internet telephony (IPTV);

[3] http://www.orange.com/english/home.php

- Voice Messaging (VM);
- High Definition television (HDTV);
- High speed WiFi (HSW);
- Email service (EMS);
- Virtual Drive (VD).

6.1.2 Our Approach

With the aim of dynamically generating customized packages each of the latter offers are interfaced by a semantic Web service (e.g., IPTVService interfaces the IPTV offer), facilitating flexible and scalable applications through the loosely coupled features of Web services. The semantic features of semantic Web services enable us not only to represent knowledge and improve expressivity levels of their functional parameters i.e., IOPEs, but also to reason about those parameters and their potential semantic connections. Here we focus on this last aspect and will provide some practical examples of our approach. The ultimate goal is to provide a correct composition of the latter Web services in regard to their semantic connections.

According to a semantic context, Figure 1.4 describes a subset of defined concepts in the $\mathcal{ALE}$ domain ontology T (305 defined concepts and 117 object properties) used to describe the domain. All input and output parameters of semantic Web services refer to concepts of the ontology T. The description Logic (DL) $\mathcal{ALE}$ is used to model our domain due to its interesting trade-off between expressivity and complexity. This scenario together with the following two use this same schema. In the following example, we consider six Web services i.e., a subset of the 35 Web services included in the real scenario:

- `AdslEligibility-`, `AdslEligibility` and `AdslEligibility+`, that from a `PhoneNum`, a `ZipCode` and an `Email` address, return respectively the `SlowNetworkConnection`[4], `NetworkConnection` and `FastNetworkConnection`[5] of the desired zone. These three Web services have been first introduced in Example 4 of Chapter 1;
- `VoiceOverIP`, that from a `PhoneNum` and a `SlowNetworkConnection`, returns the `VoIPId` of the ADSL line a Telecommunication operator needs to install the line;
- `TvOverIP`, that from a `PhoneNum` and a `FastNetworkConnection`, returns a serial number of a `VideoDecoder` required to access video over IP;
- a `LiveBox` service returns the `Invoice` of the commercial offer the user requested, depending on a `PhoneNum`, `IPAddress` and serial number of a `Decoder`.

We reduced the number of 35 services to 6 service by simply reducing the number of offer the end-user can select. This set is very flexible and can be adapted with further optional offers.

Example 65. ***(A Motivating Illustration)***
Suppose a client wants to customize and create her own internet package from a set of available technical and optional offers.

The client is then in charge of selecting the offer she is interesting in, for instance,

[4] - since the connection returned by `AdslEligibility-` is slower than `AdslEligibility+`.
[5] + since the connection returned by `AdslEligibility+` is faster than `AdslEligibility-`.

- *`AdslEligibility` to retrieve information about her connection details;*
- *`VoiceOverIP` for Internet telephony;*
- *`TvOverIP` to obtain the Television access via her internet connection;*
- *and `LiveBox` to get the invoice and technical details of her customized internet package.*

Our goal consists in retrieving a correct composition i.e.,

> A partial order amongst the selected offers (i.e., services) since the selected offers required to be ordered.

Indeed some constraints between services are in place to perform the customized internet package. For instance the service `AdslEligibility` is required to be executed first since such a service is required to obtain technical details about the internet connection the user could receive. Then the services `VoiceOverIP` and `TvOverIP` might be launched depending on the result of the `AdslEligibility` service. Finally the `LiveBox` service is in charge of aggregating information provided by the `VoiceOverIP` and `TvOverIP` services to calculate and send the final invoice to the client.

The latter partial order is defined by means of semantic constraints between services. In this scenario the semantic constraints are mainly guided by the functional parameters of services i.e., input and output parameters. Since the main idea is to customize commercial offers in an automated way, it seems conceivable to suggest a solution that computes compositions of services in a dynamic way, depending on the selected offers.

6.1.3 Open Issues and Challenges for any Telecommunication Operator

In such a scenario automation of Web service composition is a real and still an open issue, not only for France Telecom but also for any other Telecommunication operator since the number of offers i.e., Web services the user can choose is more and more increasing.

Even if this number of services is relatively reduced in our scenario, it can be conceivable that a service be offered by any Telecommunication operator or other service provider.

In this way the end-user will be able to compose its own internet package from varied services providers. Such cases will cause a duplication of services (from different sources) involved in the compositions, hence an exponentially increase of the number of service compositions. Indeed the more offers the harder the composition will be. That is why we suggest to compose automatically Web services depending on the user requirements (through the commercial offer she subscribed) and the service compatibilities (i.e., through their semantic connections).

6.2 An E-Tourism Application: The Virtual Travel Agency

This reference scenario takes some inspiration from the Virtual Travel Agency scenario (VTA) proposed in Deliverable [166] of the European project and Network of Excellence *Knowledge Web*. Here we rephrase and extend it in order to better highlight the need for composing Web services at semantic level, and the added value of doing so.

6.2.1 Motivation

The Virtual Travel Agency (VTA) is an e-Tourism service provider which offers travel booking services to the end user by using and interacting with other, more basic e-Tourism service providers.

The functionality of the VTA is that of a traditional travel agency:

- getting a request from a customer;
- dealing with different e-Tourism providers to put together an appropriate offer covering the customer request;
- arranging all the booking (and payment) with the different providers;
- and transparently offering the final trip arrangement to the customer.

While conceptually simple, this scenario, in its whole description, is quite complex; even in a very simple situation where the user only intends to consider trains and flights as possible transportation means, and where the administration only may provide credit cards and cheques as possible payment means.

6.2.2 Our Approach

In the same way as the Telecommunication scenario, the problem is mapped to a composition of semantic Web services. Therefore functional parameters of services are described by DL concepts (for instance, Trip, Itinerary, Date, TransportationMean, Price, PaymentMean, CarrierName, Time, Payment Authorization) , here in an $\mathcal{FL}_0$ (less expressive than $\mathcal{ALE}$) domain ontology (here 60 defined concepts and 19 object properties). Figure 6.1 described a sample of the ontology we used in the scenario. We briefly describe five Web services among the 45 services involved in the implemented scenario.

- Two `Travel information` services: one regarding trains, the other flights; each of these, receiving a request for a specific destination, returns a possible ticket to be bought, completed with its price;
- The `Administration` service, which represents the employer's administration; once received a request for a work trip and a possible ticket, it evaluates it, and if the budget is sufficient and the request is consistent, provides either a cheque or a credit card with which the ticket can be booked;
- Two `Payment` services, one handling credit cards, the other handling cheques.

Here we assume that the available e-Tourism providers should be located dynamically by the VTA, with no need for prior agreements, and that the business process of the VTA should be composed dynamically based on the request received and the available providers.

Example 66. ***(A Motivating Illustration)***
The customer wants to make a trip to a given location (e.g., Rennes, France) for a given period of time (e.g., staying there from August 10 to August 15). The customer sends his request to the VTA, which has to build a package including a travel to/from Rennes and an accommodation for all the nights spent in Rennes. Clearly, the hotel has to be booked according to the flight (i.e., if the flight arrives on August 9, then the hotel has to be booked from August 9).

$$
\begin{array}{l}
Location \sqsubseteq \top,\ Date \sqsubseteq \top,\ Time \sqsubseteq \top,\ CarrierName \sqsubseteq \top \\
PaymentMean \sqsubseteq \top,\ TransportationMean \sqsubseteq \top,\ Price \sqsubseteq \top \\
Trip \sqsubseteq \forall from.Location \sqcap \forall to.Location \sqcap \forall when.Date \\
\qquad \sqcap \forall by.TransportationMean \\
Ticket \sqsubseteq \forall from.Location \sqcap \forall to.Location \sqcap \forall when.Date \\
\qquad \sqcap \forall time.Time \sqcap \forall with.CarrierName \sqcap \forall cost.Price \\
BookedTicket \equiv Ticket \sqcap \forall paid\_with.PaymentMean \\
PaymentAuthorization \sqsubseteq \forall with.PaymentMean \sqcap \forall cost.Price \\
Train \sqsubseteq TransportationMean \\
Flight \sqsubseteq TransportationMean \\
TrainTrip \equiv Trip \sqcap \forall by.Train \\
FlightTrip \equiv Trip \sqcap \forall by.Flight \\
TrainTicket \sqsubseteq Ticket \\
FlightTicket \sqsubseteq Ticket \\
CreditCard \sqsubseteq PaymentMean \\
Check \sqsubseteq PaymentMean \\
CreditCardAuthorization \sqsubseteq PaymentAuthorization \sqcap \forall with.CreditCard \\
CheckAuthorization \sqsubseteq PaymentAuthorization \sqcap \forall with.Check
\end{array}
$$

Figure 6.1: Terminology $\mathcal{FL}_0$ for the E-Tourism Use Case.

The VTA should take care of locating the necessary tourism service providers (e.g., suitable flight providers for the trip, hotels in Rennes...) and contact them. Finally, a suitable offer will be returned to the customer and upon acknowledgement either both the accommodation and the travel shall be booked or none, which requires a weak form of transactionality for the composed service.

6.2.3 Open Issues and Challenges

In such a scenario automation of Web service composition is an open issue, not necessarily for Telecommunication operators but more for service providers, as well as contents providers. In the same way as the Telecommunication scenario the more services the more complex is the composition process. Moreover expressivity of services (i.e., complexity of their functional descriptions) has a crucial impact on the composition performance (in terms of computation time).

These challenging issues are addressed with an automated process of composition.

In addition to the latter issue, issues related to *trust and reputation management* (e.g., by regulating the service requester's access to the service provider), *negociation*, *security* need to be studied in further details.

6.3 An E-HealthCare Application

This last introduced scenario takes place in the e-HealthCare area. In such a context we focus on the medical devices. A medical device is an object which is useful for diagnostic or therapeutic

purposes. The medical devices have become an increasingly important health care area in relation to their impact on health and health care expenditure. The sector covers some 8000 types of products, ranging from simple bandages and spectacles, through life maintaining implantable devices, equipment to screen (e.g., high-tech sphygmomanometer) and diagnose disease and health conditions, to the most advanced diagnostic imaging and minimal invasive surgery equipment.

In this scenario we are interested on the latter sophisticated medical devices i.e., devices that return some (medical, analysis) results depending on some parameters.

Moreover we assume (wrongly now but rightly in the very next future) that such devices are all electronic, not expensive and easy to operate by any end-users (such as patients their family).

6.3.1 Motivation

The idea behind this scenario is as follows "*Providing a distant follow-up of patients*". By providing such a new way of follow-up we aim at reducing the extra number of consultations, examinations, medical check-ups and consequently their price. Indeed a long-standing clinical observation in hospital is no longer a realistic issue for cost reasons since the elderly.

Towards such issues we focus on providing an automated way to order and compose medical devices, given a goal and some requirements. This will automate inter-operation between medical devices.

6.3.2 Our Approach

In the same way as the Telecommunication scenario, we suggest to interface devices with Web services. This ensures to easily work in heterogeneous environments and various domains of application. Thus telemedical collaborations are possible through the Web service paradigm. A solution of such a problem consists in implementing a composite and value-added Web service that can automate the patient follow-up by a reliable Web service interoperation, hence a long distance follow-up. Therefore the problem is turned into a problem of semantic Web service composition. Functional parameters are then described by DL concepts. Figure 6.2 describes a subset of defined concepts in the $\mathcal{ALE}$ domain ontology $\mathcal{T}$ (105 defined concepts and 37 object properties) used to describe the domain. In the following example, we illustrate three Web services of the set of 12 Web services included in the real scenario:

- A `sphygmomanometer` service is a Web service that interfaces a blood pressure meter, used to measure blood pressure of a given patient;
- A `thermometer` service is a service that interfaces a thermometer which is used for measuring human body temperature;
- A `Blood glucose monitor` service is used for testing the concentration of glucose in the blood of a patient;

Example 67. ***(A Motivating Illustration)***
*Suppose a patient that accepts a long distance follow-up of this disease at home. In such a case all required medical devices together with their Web services interfaces are installed at her home. Once the material is ready to be used, the patient can be "*connected*" to these devices through sober sensors. Therefore the health of the patient can be frequently sensed by medical devices. The result of this different sensors is sent to hospital, for instance to the E-mail address of the assigned physician. In case of critical values of the blood pressure's, body temperature's and*

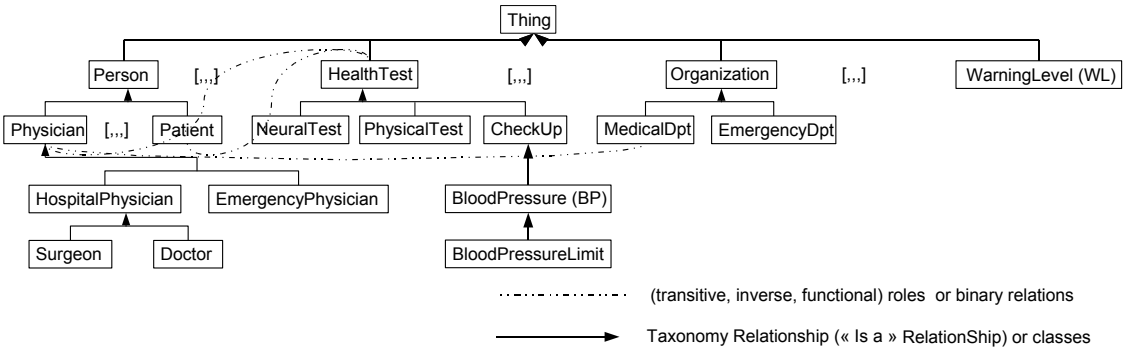


Figure 6.2: A Sample of an E-healthcare Domain Ontology $\mathcal{T}$.

glucose rate's patient the emergency services are contacted through, one more time, a relevant Web service.

The distance follow-up is concretized by an inter-operation of medical devices (through their Web service interfaces) wherein i) the patient and some nurses are required to correctly use devices and ii) physicians can interpret results of the inter-operation of devices (e.g., some complex graphics).

6.3.3 Open Issues and Challenges

Open issues and challenges are same as the first two scenarios. However one more significant problem could arises i.e., the end user confidence in these new technologies is crucial to its success.

6.4 Many Other Potential Applications

The spectrum of applications seems very wide. Indeed we could easily give some nowadays examples such as: enterprise portals, Capitalization of knowledge, E-commerce, E-learning, E-work, E-business, E-health, E-government and E-administrations, natural language processing and machine translation, information retrieval, integration data and services, social networks, systems recommendations and collaborative filtering, extraction of actionable knowledge, economic intelligence and others.

6.5 Conclusion

In this Chapter three scenarios in use have been presented. The first scenario operates in a Telecommunication domain whereas the two others acts respectively in area of E-Toursim and E-Healthcare. Each of these use cases requires automated composition of Web services in a semantic context. The semantic expressivity and number of Web services vary from a scenario to another i.e.,

i) one in the Telecommunication domain (the extended version of the motivating scenario in Section 6.1) where the number of potential Web services ($\#S^*_{Ws}$) is 35 and the TBox of the $\mathcal{ALE}$ Ontology consists of 305 defined concepts and 117 object properties;

ii) another in the E-Tourism domain where $\#S^*_{Ws}$ is 45 and the TBox of the $\mathcal{ALE}$ Ontology consists of 60 defined concepts and 19 object properties;

iii) and finally one in the E-HealthCare domain where $\#S^*_{Ws}$ is 12 and the TBox of the $\mathcal{ALE}$ Ontology consists of 105 defined concepts and 37 object properties.

All these scenarios have been tested with our composition approach proposed in Chapters 4 and 5. Results of experiments are presented in Chapter 7. In the next Chapter we draw some evaluation results that show high efficiency and effectiveness of our composition model.

Chapter 7

The Composition Tool: Implementation & Experiments

In this Chapter we discuss the prototype tool that we developed to compute automated semantic Web service compositions and its optimal composition in our framework.

In addition to a reference architecture, we give an evaluation of its main components by analyzing their experimental results on different levels of scenarios (i.e., scenarios in use from Chapter 6 and random scenarios).

The components we plan to study in more details are components that implement our composition model i.e., works presented in the Part II i.e.,

- Chapter 3 and its *SLM model*;
- Chapter 4 and its two *composition approaches*;
- Chapter 5 and its *optimization process*.

The remainder of this Chapter is described as follows. Section 7.1 presents the reference architecture used to achieve optimal composition of Web services. Moreover a detailed view of its core components and their implementation is presented. Section 7.2 describes some experimental results obtained by running our reference architecture on the three scenarios of Chapter 6. Roughly speaking this section studies the impact (in terms of computation time performance) of each (innovative or not) component on the reference architecture. Section 7.3 mainly focuses on the computation time performance and scalability of the two composition approaches and the optimization process in more general scenarios. Moreover this section sketches the main features that characterize each of the latter components. Finally Section 7.4 draws some concluding remarks on the suggested architecture, their innovative components and the experimental results.

7.1 Architecture and Implementation

Along this Ph.D report we studied functional level based semantic Web service composition. Starting from an initial set of relevant Web services, such a level of composition aims at selecting and inter-connecting web services according to a goal to achieve.

To this end we focus more particularly on two different approaches to achieve such a level composition. Depending on:

- the level of Web service description (**Requirement** $R^{Service}_{Expressivity}$) and their interfaces to standard proposals (**Requirement** $R^{Service}_{Applicability}$);
- the composability criteria (**Requirement** $R_{Composability}$) pertaining automatic composition of services, described in terms of input, output parameters and preconditions, effects. Such citeria are used to relate some Web services in a composition i.e., **Requirements** $R^{Semantic}_{Composability}$ and $R^{Causal}_{Composability}$;

the two automated (**Requirement** $R^{Composition}_{Automation}$) and flexible (**Requirement** $R_{Flexibility}$) approaches return a partial order of Web services with different expressivity of their computed compositions (**Requirement** $R^{Composition}_{Expressivity}$). From this order of services a component which is responsible of optimization (**Requirement** $R_{Optimization}$) is required to compute an optimal composition in terms of *semantics* of its links. Finally, as we will see in this Section, the optimal composition is rendered in a standard BPEL4WS composite service (**Requirement** $R^{Composition}_{Applicability}$).

All these different components are presented, ordered, implemented (as core components) and tested in our high level reference architecture. This architecture is depicted in Figure 7.1.

In a nutshell, the reference architecture consists of the following components:

- a *Repository of Semantic Web Services* that contains available services together with their (syntactical and semantic) functional description;
- a *Domain Ontology* that contains descriptions of concepts, instances, respectively defined in its TBox and Abox;
- a *Service Goal* which will be realized by the composition result;
- a *Service Discovery and Selection* component that discovers the most relevant Web service to compose and satisfy the service goal;
- a *Semantic Reasoning* component that computes semantic links and value robustness, validity, matching quality and common description rate of Web service compositions;
- a *Causal Laws Reasoning* component in order to reason on causal laws between Web services;
- a *Functional Level Composition* component that can achieve composition with two different methods, depending on the composability criteria;
- a component dedicated to the *Composition Optimization*;
- a final component responsible of *BPEL Rendering* to easily implement, execute, reuse the resulting composition of Web services.

In the rest of this section we present in more details these latter components and their implementation.

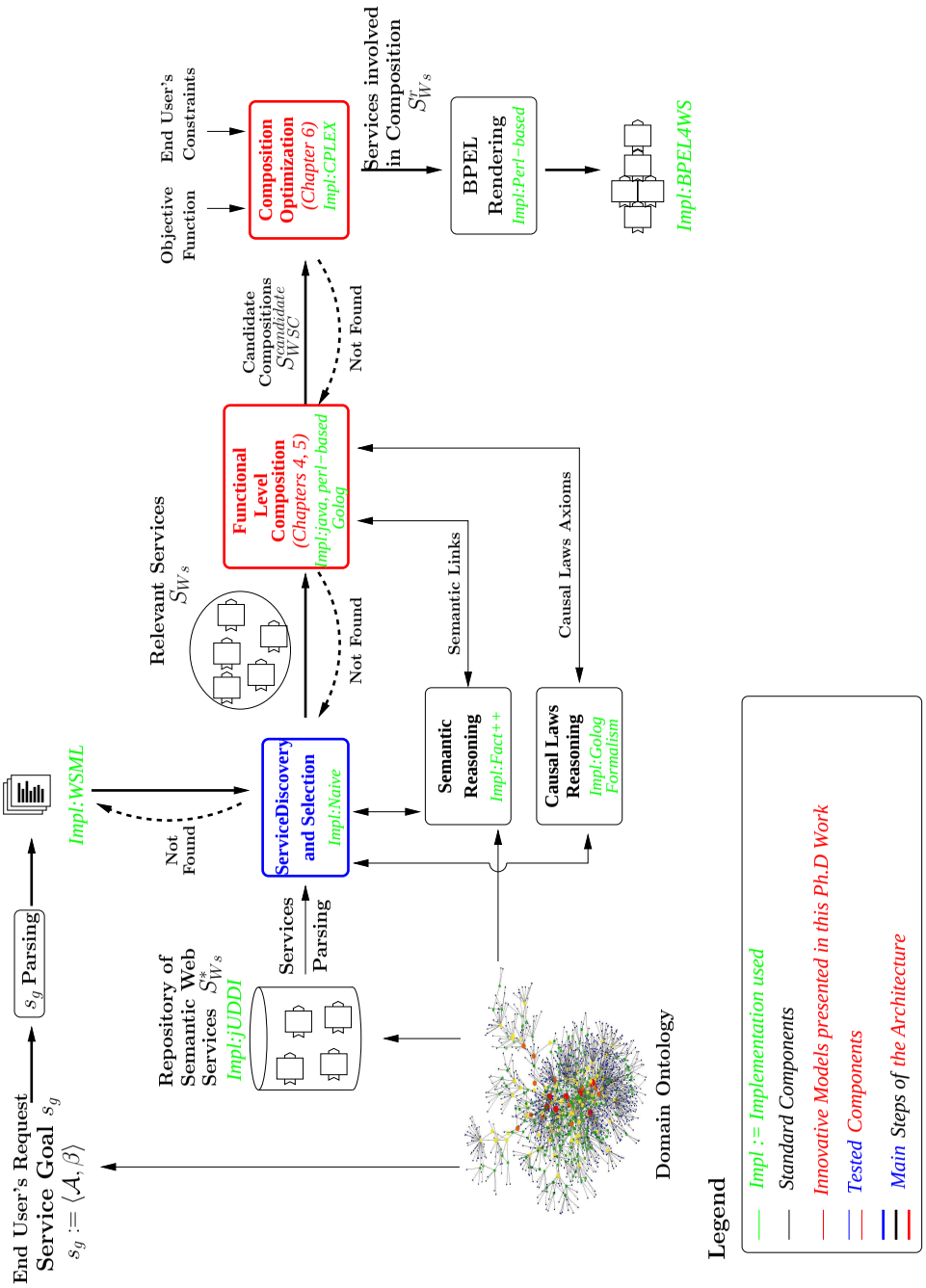


Figure 7.1: The Reference Architecture, Its Innovative Components and their Implementation (A No Detailed Version).

7.1.1 Repository of Semantic Web Services

Description

According to the reference architecture depicted in Figure 7.1, an important and required component is the *Repository of semantic Web services*. This repository is defined as the set S^*_{Ws} of services. This repository of services implements our different scenarios, and can be seen, therefore, as an advanced version of UDDI.

In the suggested framework each entry of the UDDI registry represents a service in terms of both its syntactic interface through a WSDL document, and its semantic description[1], which can be expressed in any language that allows to express semantics. In this direction we focus on semantic and syntactic-based service descriptions. We recall that in our framework the focus is on semantics that a service can express.

Input of the Component: *Some services and a domain ontology.*
Output of the Component: *The set S^*_{Ws} of services.*

Implementation

The repository of semantic Web services has been implemented by means of jUDDI i.e., an open source Java implementation of the Universal Description, Discovery, and Integration (UDDI) specification for Web Services. In the prototype implementation semantic Web services are defined through the WSML specification. As an example, the functional description (through its WSML interface) of service `AdslElegibility`* S^*_a can be consulted in Figure 7.2). In the previous WSML illustration, the service is described by means of its input, output parameters and preconditions, effects (**Requirement** $R^{Service}_{Expressivity}$).

7.1.2 Domain Ontology

Description

The *Domain Ontology* contains descriptions of concepts (in its TBox, Figure 1.4) used to semantically annotate Web services in their semantic forms. Moreover the domain ontology contains instances which are defined in its Abox.

Input of the Component: *None.*
Output of the Component: *Semantic description of the domain.*

Implementation

In this prototype, we achieve semantic description of Web services by means of WSMO.

[1] More specifically, we exploit the UDDI tModel data type to provide the technical specifications of services

7.1.3 Service Goal s_g

Description

In our prototype, the end-user is in charge of specifying her goal service s_g in terms of a pair $\langle \mathcal{A}, \beta \rangle$. Roughly speaking s_g (i.e., an abstract services defined in terms of their input and output parameters) is described completely and sufficiently by instances and concepts chosen from the ontology.

In particular the terms $\mathcal{A}$ refers to instances occurring in the ABox of the domain ontology and β is defined as a subset of concepts in the TBox $\mathcal{T}$ of this same ontology. The parameters of s_g may refer to concepts in the TBox $\mathcal{T}$ of the domain ontology, or can use some extra concepts that will extend the initial TBox $\mathcal{T}$.

The successful composition of Web services require instances of $\mathcal{A}$ as input parameters, and concepts of β as output parameters. In our approach, the goal is to retrieve some instances of β by achieving a composition of Web services.

Input of the Component: *Domain Ontology.*
Output of the Component: *Service Goal s_g.*

Implementation

From this, the goal s_g is translated into a WSML document (through a s_g **Parsing** step in Figure 7.1), expressed by means of its functional parameters. Therefore both the services in the repository and the goal service s_g are described at semantic level, using the same WSML formalism.

7.1.4 Service Discovery and Selection Component

Description

Given a set of available (not necessarily relevant) Web services in the repository and a service goal to achieve, the composition component requires an important first component i.e., the *Web service Discovery and Selection* component. This component is in charge of retrieving relevant Web services S_{Ws} depending on the composition goal s_g (provided by end-user). The output set S_{Ws} of this component is defined as a subset of the initial set of available Web services S^*_{Ws}.

Since the discovery component aims at comparing the semantics of functional parameters, some closed interactions with the `Semantic Reasoning Component` are performed.

Input of the Component: *The service goal s_g, the set S^*_{Ws} of services.*
Output of the Component: *A relevant set S_{Ws} of services.*

Implementation

This discovery component is implemented in our reference architecture by a naive algorithm. Roughly speaking the set S^*_{Ws} is discovered by means of a breadth first forward search. At each

```
wsmlVariant _"http://www.wsmo.org/wsml/wsml-syntax/wsml-rule"
namespace{ _"https://sws-orangeLabs.elibel.tm.fr/Sm3e-Pro/ontologies/internetPack#",
        dc _"http://purl.org/dc/elements/1.1#",
      foaf _"http://xmlns.com/foaf/0.1/",
      wsml _"http://www.wsmo.org/wsml/wsml-syntax#",
       loc _"http://www.wsmo.org/ontologies/location#"}

/************* SaStar ONTOLOGY ***************/
ontology _"https://sws-orangeLabs.elibel.tm.fr/Sm3e-Pro/ontologies/SaStar_Ontology"
       nfp
          dc#title hasValue "WSML ontology of SaStar i.e., AdslEligibilityStar"
          dc#subject hasValue "InternetPackage"
          dc#description hasValue "Fragments of the InternetPackage Ontology"
          dc#contributor hasValue
             { _"https://sws-orangeLabs.elibel.tm.fr/Sm3e-Pro/~fael8534/foaf.rdf"}
          dc#date hasValue _date(2007,08,28)
          dc#format hasValue "text/html"
          dc#language hasValue "en-US"
          dc#rights hasValue _"https://sws-orangeLabs.elibel.tm.fr/privacy.html"
          wsml#version hasValue "$Revision: 1.1 $"
       endnfp

concept ZipCode
        zone ofType Country
concept Email
        type ofType Any
concept PhoneNum
        zone ofType Country
concept NetworkConnection
        type ofType Any
        netSpeed ofType Speed
        nonFunctionalProperties
                dc#description hasValue "concept of a NetworkConnection"
        endNonFunctionalProperties

/************** WEB SERVICE ****************/
WebService _"https://sws-orangeLabs.elibel.tm.fr/Sm3e-Pro/services/SaStar"
importsOntology
           _"https://sws-orangeLabs.elibel.tm.fr/Sm3e-Pro/ontologies/IPDO.wsml"
capability _"https://sws-orangeLabs.elibel.tm.fr/Sm3e-Pro/Eligibility#SaStarCap"

sharedVariables {?pn, ?zc, ?em}

//Preconditions in WSMO refer to Input parameters in the Web Service Specification
precondition  definedBy
                ?pn memberOf PhoneNum
                and
                ?zc memberOf ZipCode
                and
                ?em memberOf Email.

//PostConditions in WSMO refer to Output parameters in the Web Service Specification
postcondition definedBy
                ?nc memberOf NetworkConnection.

//Assumptions in WSMO refer to Preconditions in the Web Service Specification
assumption definedBy
                ?pn [zone ofType France]
                and
                ?zc [zone ofType France]
                and
                ?em [type ofType Valid].

//Effects in WSMO refer to Postconditions in the Web Service Specification
effect definedBy
                ?nc [type ofType Valid].
```

Figure 7.2: WSML Description of the AdslEligibility* Service S_a^*.

step, the set of all semantic output parameters produced by the services discovered so far is passed as input to the UDDI registry search, so to discover every service that can be possibly activated by combining results of previously discovered services. Service collection stops either when the sets of output parameters cover those in the output portion of the service goal, or when no new service is returned by UDDI search, i.e., when a fix point is reached.

7.1.5 Semantic Reasoning Component

Description

The reference architecture is completed with a *Semantic Reasoning* component (required by **Requirement** $R^{Semantic}_{Composability}$), which provides a vital infrastructural support to two components of the architecture i.e., i) the *Service Discovery and Selection Component* and ii) the *Functional Level Composition* component.

The main function of this component is to infer some properties on functional input and output parameters (defined as concepts of the TBox $\mathcal{T}$) of semantic Web services. The *reasoning* component can check, for instance, satisfiability or subsumption of Web service parameters by means of a DL reasoner. Therefore semantic links between Web services can be *semantically* valued.

Input of the Component: *Domain ontology, (semantic descriptions of) input and output of services.*
Output of the Component: *Semantic links between services.*

Implementation

In our approach we adopt the WSMO4J implementation for WSMO parsing of the domain ontology and make use of the open source Fact++ DIG reasoner[2] [92] (Pellet [189] or RacerPro [82] can be used as well to compute standard reasoning and the MAMAS-tng[3]) to compute non standard reasoning such as Concept Abduction [48]. Concept Difference is computed by means of an internal France Telecom component. The power of such a component is therefore crucial to the performance of the overall architecture.

7.1.6 Causal Laws Reasoning Component

Description

The *Causal Laws Reasoning* component aims at first relating effects of services and preconditions of other Web services (i.e., causality relationships). In other words this component discovers *possible* Web services i.e., services with no open preconditions (see Section 4.2 for more details). Secondly the *Causal Laws Reasoning* component required the complex relationships between Web services to relate some of their parameters.

This component has a direct interaction with the *Functional Level Composition* component in cases of i) Web services described by their inputs, outputs, preconditions, effects and ii) causal

[2] http://owl.man.ac.uk/factplusplus/
[3] http://dee227.poliba.it:8080/MAMAS-tng/DIG

laws **Requirement** $R^{Causal}_{Composability}$ used as a composability criterion.

Input of the Component:	*Preconditions and effects of services, complex relationships between services.*
Output of the Component:	*Causal laws between services.*

Implementation

The implementation of this component is based on a Golog formalism, especially for the axiomatization of the causal laws in $s_{sl}Golog$.

7.1.7 Functional Level Composition Component

The `Functional Level Composition Component` is the core module of the reference architecture. This component is responsible of the computation of Web service compositions. From a set of relevant Web services S_{Ws}, it computes a set of compositions $S^{candidate}_{WSC}$ that achieve the service goal s_g. Two different composition approaches can be performed depending on the level of Web service description (Requirement **Requirement** $R^{Service}_{Expressivity}$).

Input of the Component:	*A relevant set S_{Ws} of services, the service goal s_g.*
Output of the Component:	*A set of compositions $S^{candidate}_{WSC}$.*

Semantic Links based Web Service Composition

a) Description:

On the one hand the *Functional Level Composition* component can be achieved by defining the semantic links as the only composability criterion (Figure 7.3).

In such a case the component which is responsible of this composition consists in three important components i.e., the *SLM Construction* component , the Ra_4C component and the *robustness verification* component (Algorithm 3 in Chapter 4). The different processes of these components have been first presented in Section 4.1.

First of all the *SLM Construction* component elaborates the SLM (algorithm 1) of the composition problem by considering the service goal s_g and the set of Web service S_{Ws} returned by the *Service Discovery and Selection* component. The *SLM Construction* component is related to the *Semantic Reasoning Component* since its construction requires valuation of semantic links.

Then the Ra_4C component is responsible of computing correct, complete, consistent and (optionally robust) composition of Web services according to the SLM of the domain and the service goal s_g.

Finally the robustness feature of Web service composition is ensured by means of the *Robustness Verification* component together with the *Semantic Reasoning* component. In this case a non standard reasoning such as Concept Difference or Concept Abduction is required.

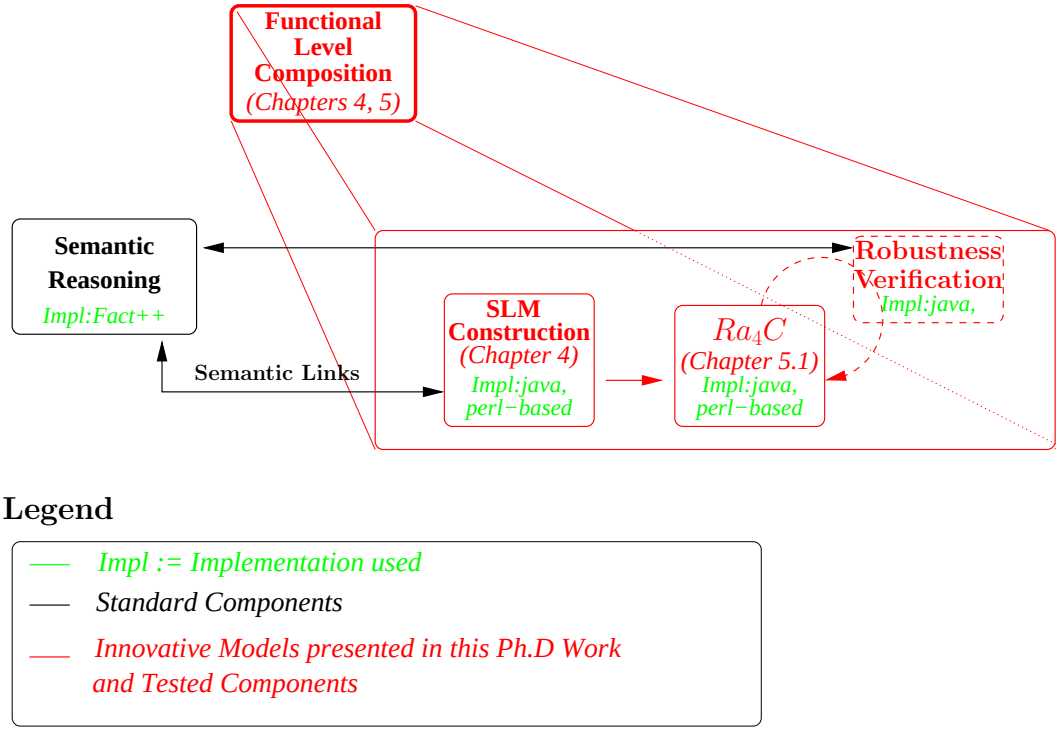


Figure 7.3: Architecture of the Semantic Links based Web Service Composition Component.

a) Implementation:

This component has been implemented using Java, C++, Prolog and Perl.

Semantic Links and Causal Laws based Web Service Composition

a) Description:

On the other hand the *Functional Level Composition* component can be achieved by defining the semantic links and causal laws as both composability criteria (Figure 7.4).

In such a case the component which is responsible of this composition consists in two important components i.e., the $s_{sl}Golog$ component and the *Backward Chaining* component. The different processes of these components have been first presented in Section 4.2.

In this composition approach the composition problem is first modelled in the $s_{sl}Golog$ language i.e., Web services are modelled as actions, axioms related to the initial situation, causal laws, preconditions and semantic links are characterized.

Then, an adaptation of the *Backward Chaining* based planner for $s_{sl}Golog$ is interacting with axioms of the composition problems to compute conditional compositions of Web services. This will ensure to compute compositions with semantic links together with causal laws.

$s_{sl}Golog$ is interacting with the *Semantic Reasoning Component*, especially to compute values (i.e., semantic match types) of semantic links between Web services. Note that robustness of Web service composition is ensured in case the set of axiom of $s_{sl}Golog$ contains the $Robust_{matchType}$ axiom instead of the $Valid_{matchType}$ axiom. Moreover $s_{sl}Golog$ is interacting with the *Causal Laws reasoning* component, especially to compute causal laws between Web services.

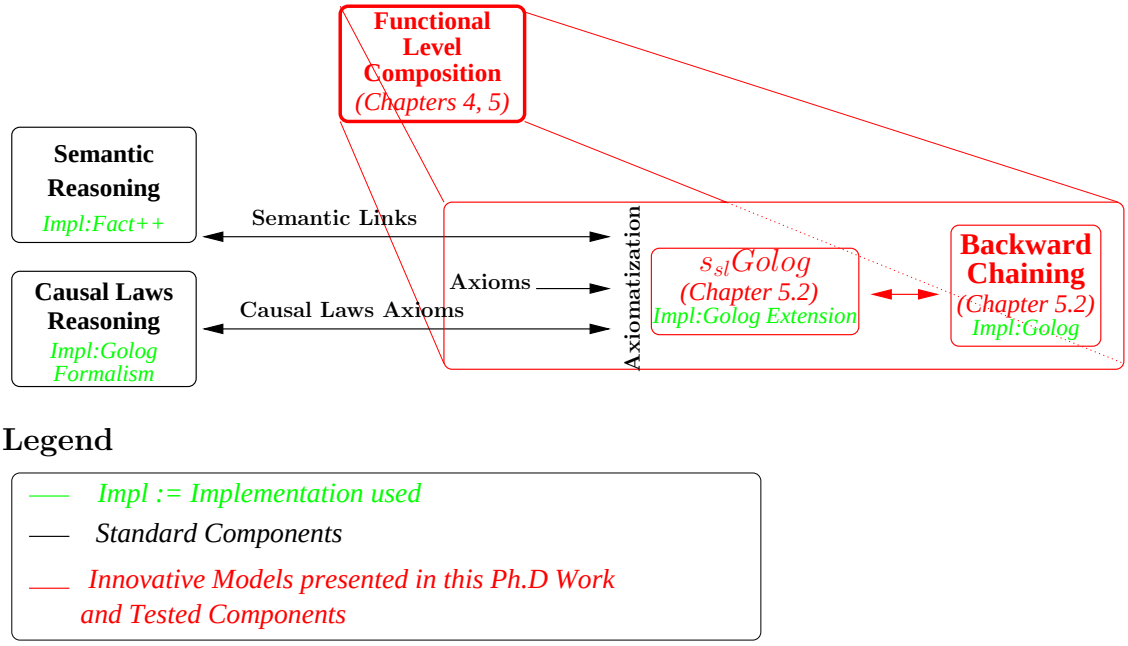


Figure 7.4: Architecture of the Semantic Links and Causal Laws based Web Service Composition Component.

b) Implementation:

The implementation of $s_{sl}Golog$ is based on an extension of Golog (in Eclipse Prolog) wherein axioms related to *valid* and *robust* actions are defined. Moreover axioms related to *General Conditional Action Trees* (i.e., *mb_on*, *switch*) have been implemented in $s_{sl}Golog$. The latter axioms ensures to compute conditional compositions of Web services.

7.1.8 The Composition Optimization Component

Description

Since the *Functional Level Composition* component aims at computing a set of compositions $S_{WSC}^{candidate}$ that achieved the same service goal, a component responsible of composition selection is required to prune the latter set and then obtain a unique composition of services. The latter composition consists of Web services defined in the new set S_{Ws}^{r}. Both components described in Section 7.1.7 can be used to compute the set of candidate Web service compositions $S_{WSC}^{candidate}$.

This component makes sure the end-user has the most optimal composition (**Requirement** $R_{Optimization}$). Here, the optimization is oriented by the semantic quality of compositions i.e.,

quality of their semantic links. Thus we assumed that robustness, common description rate and matching quality of each semantic link have been inferred by means of the *semantic reasoning* component (in a pre-processing step).

Input of the Component: *A set of compositions* $S_{WSC}^{candidate}$, *an obejtive function, some end-user constraints.*
Output of the Component: *An optimal composition consisting of a set of services defined in* S_{Ws}^{r}.

Implementation

The reference architecture interfaces this components by an optimization problem through an Integer Linear Programming problem. Therefore, first the Integer Linear Programming model formulation is computed, and the optimization problem is solved by running CPLEX, a state of the art integer linear programming solver based on the branch and cut technique [4][204].

7.1.9 The BPEL Rendering Component

Description

Once a correct, consistent, complete, optimal (and optionally robust) composition of semantic Web services is returned by the *Composition Optimization* component, the resulting composition is rendered in a BPEL4WS [10] composite service.

Input of the Component: *A composition of Web services.*
Output of the Component: *A BPEL4WS description of the composition.*

Implementation

This rendering component is required to model a Web service composition in a standard format (Requirement $R_{Applicability}^{Composition}$), which can be easily deployed and executed. The BPEL4WS output file consisted of sequences, concurrent and conditional branches of Web services, non determinist choice of Web services (Requirement $R_{Expressivity}^{Composition}$) and assignments between parameters of services (in case of semantic links).

7.1.10 Synthesis

Here, we presented the reference architecture (Figure 7.5) wherein our innovative components have been integrated with standard components.

On the one hand each standard component of the reference architecture (i.e., the *repository of semantic Web services*, the *Service Discovery and Selection* component, the *semantic reasoning* component and the *BPEL rendering component*) relies on powerful state-of-the art technologies

[4]LINDO API version 5.0, Lindo Systems Inc. http://www.lindo.com/

and tools. On the other hand *Functional Level Composition* is performed by means of our approach (an adapted regression based approach; an extension of Golog), as well as the *Composition optimization*.

The remaining code, which includes the various algorithms developed for the architecture as well as the interfacing to the external tools, is realized in Java and ANSI C, for maximum portability. To enable the direct end to end utilization of our architecture, we also included a component responsible of final automated deployment. To this end we assume the presence of an Active Web Flow engine running over a local Apache Tomcat platform. Table 7.1 summarizes the Requirement $R_{Applicability}$.

Requirement R_i	Details	Chapter 7
$R_{Applicability}$	$R^{Service}_{Applicability}$	Applicable to the OWL-S service profile, WSMO service capability, SWSO Inputs/Outputs, SA-WSDL sepecification, depending on the composition approach.
	$R^{Composition}_{Applicability}$	BPEL4WS through the BPEL Rendering Component.

Table 7.1: Table of Requirements supported by Chapter 7.

While we described our architecture instantiation supporting semantic and specific requirements and implementations of services, we stress that the architecture design is language-independent. Indeed different instantiations are also possible as well.

In the following section all components involved in the reference architecture depicted in Figure 7.1 have been evaluated in some practical experimentations.

7.2 Results and Empirical Evaluation on Three Scenarios in Use

This section describes some experimental results obtained by running our implemented prototype system (see our reference architecture in Figure 7.1) on the three scenarios of Chapter 6 i.e., the Telecommunication, the E-Tourism and the E-Healtcare scenarios. More specially, this section studies and evaluates the impact and difference, in terms of computation time performance, of each (innovative or not) component on the reference architecture.

7.2.1 Context of Evaluation: Scenarios, Web Services and System Configuration

Main Features of Scenarios in Use

First of all, we remind the reader the main features of the tested France Telecom scenarios:

i) the *Telecommunication* scenario is defined by

- a number of potential Web services ($\#S^*_{Ws}$) equal to 35;
- a TBox of the $\mathcal{ALE}$ Ontology with up to 305 defined concepts and 117 object properties;

ii) the *E-Tourism* scenario is defined by

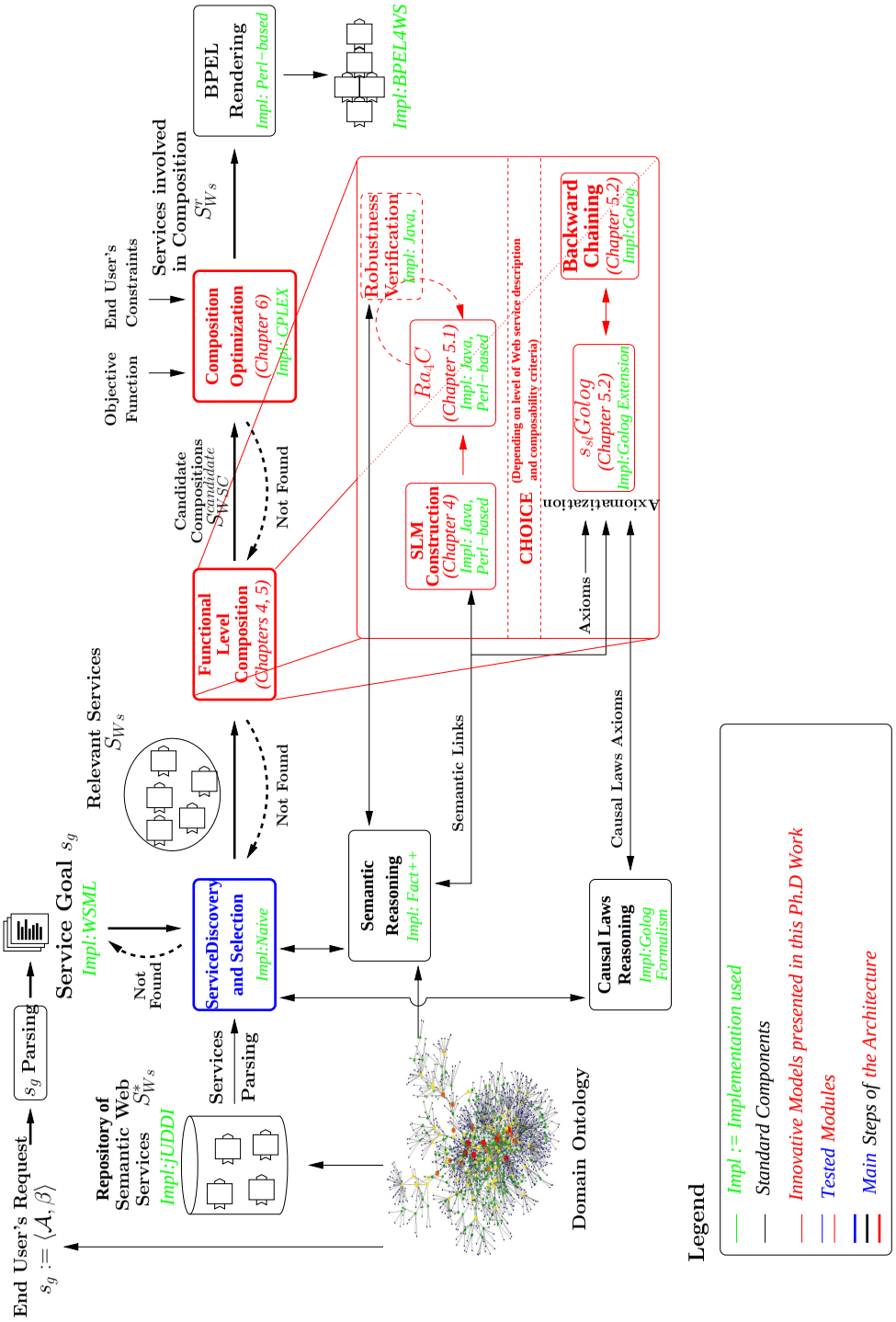


Figure 7.5: The Reference Architecture, Its Innovative Components and their Implementation (A Detailed Version).

- a number of potential Web services ($\#S^*_{Ws}$) equal to 45;
- a TBox of the $\mathcal{ALE}$ Ontology with up to 60 defined concepts and 19 object properties;

iii) the *E-HealthCare* scenario is defined by

- a number of potential Web services ($\#S^*_{Ws}$) equal to 12;
- a TBox of the $\mathcal{ALE}$ Ontology with up to 105 defined concepts and 37 object properties;

Here we describe the input parameters of each scenario. Therefore, we especially focus on **features about Web services** of the scenarios:

- $\#S^*_{Ws}$. This cardinal is the number of available Web services for a given composition problem. This number is given by the *Repository of Web Services*;
- #InputMax. This cardinal refers to the maximum number of parameters used to describe functional input parameters of Web services. In the considered scenarios, Web services in S^*_{Ws} have at most three input parameters, and at least one input parameter;
- #OutputMax. This cardinal refers to the maximum number of parameters used to describe functional output parameters of available services. In the considered scenario, Web services in S^*_{Ws} have at most three output parameters, and at least one output parameter.

Moreover, since the composition problem is depending on a service goal $s_g := \langle \mathcal{A}, \beta \rangle$, we also focus on **features about this service goal**:

- $\#\mathcal{A}$. This denotes the number of instances used to define the input parameters of the service goal s_g;
- $\#\beta$. This denotes the number of concepts used to define the output parameters of the service goal s_g.

The context and details of the three scenarios in use are summarized in Table 7.2.

Area	Main Parameters	Description of Main Parameters	Application Domain		
			Telecom $\#\mathcal{T} := 305$	**E-Tourism** $\#\mathcal{T} := 60$	**E-HealthCare** $\#\mathcal{T} := 105$
Context	**Web services**	$\#S^*_{Ws}$	35	45	12
		#InputMax	[1;3]	[1;3]	[1;3]
		#OutputMax	[1;3]	[1;3]	[1;3]
		Precondition Axioms	4	4	4
		Effect Axioms	4	4	4
	Goal s_g	$\#\mathcal{A}$	3	2	2
		$\#\beta$	1	1	1

Table 7.2: Context and Details of Scenarios in Use.

Web Service Description

The three previous scenarios have been tested by our implemented prototype system with two different levels of Web service description. Such a consideration ensures to compare results of the two *Functional Level Composition* approaches using the same reference architecture.

Therefore, on the one hand we assume that Web services used input and output parameters to describe their functionality. The composability criteria of this composition problem are the semantic links. On the other hand we assume that Web services used both input, output parameters and preconditions, effects. In such a case we assume that compositions of service are computed by means of both semantic links, and causal laws between services and their parameters.

System Configuration

In this set of experiments the PC used for running the prototype system had the following configuration: Intel(R) Core(TM)2 CPU, 1.86GHz with 512 RAM. The PC runs Linux-gnu (2.6.12-12mdk) and Java 2 Edition v1.5.0_11.

7.2.2 Experimental Results

Since our experimentation use the same *Service Discovery and Selection* component and operates on two levels of Web service description, we suggest to study the computation time performance of the following processes:

- *Service Discovery and Selection* process;
- *semantic links based web service composition* process;
- *semantic links and causal laws based web service composition* process

Service Discovery and Selection

In addition to the evaluation of the *Service Discovery and Selection* component, the main intermediate result $\#S_{Ws}$ is valued and given before the composition process of each considered scenario:

- $\#S_{Ws}$. This is the number of relevant Web services returned by the *Service Discovery and Selection* process. This set is parsed and used by the SLM construction component.

The Table 7.4 sketches the main results concerning the computation time performances of the *service discovery and selection* process on the three scenarios in use.

Area	Process	Description of Process	Application Domain		
			Telecom $\#\mathcal{T} := 305$	**E-Tourism** $\#\mathcal{T} := 60$	**E-HealthCare** $\#\mathcal{T} := 105$
Service Discovery and Selection	**Discovery (ms)**		**48.1**	**54.2**	**29.5**
		$\#S_{Ws}$	14	22	9

Table 7.3: Service Discovery and Selection on Scenarios in Use.

The *Service Discovery and Selection* process takes a negligible time i.e., below 0.1 second in two thirds of scenarios.

The computation time performance is not very high since the number of available Web services is at worst 35 i.e., no complex problem of discovery.

Semantic Link based Web Service Composition

Here, we specially focus on compositions wherein semantic links are involved (Sections 4.1). Therefore we investigate on the computation time of processes achieved by the following components:

- the `SLM Construction` and *Semantic Reasoning* components (Chapter 3);
- the Ra_4C component (Section 4.1);

- the Ra_4C component coupled with *Robustness Verification* (Section 4.1.6);
- the component responsible of the *Composition Optimization* process (Chapter 5).

Since the FLC based approach operates with an SLM of a given domain, we also focus on **information and parameters that characterize any SLMs**:

- #Rows. This represents the number of rows of the SLM;
- #Columns. This represents the number of columns of the SLM;
- Filling Rate Fr (%). This rate is given by $\frac{\#\{NonEmptyEntryset\}}{\#Rows \times \#Columns}$;
- $\overline{m}_{i,j}$. This is defined as the average of elements by non empty entry in the SLM;

In addition we characterized **the main parameter of the Ra_4C composition process**

- the number of candidate compositions returned by the composition process i.e., $\#S_{WSC}^{candidate}$. These candidate compositions are correct, complete and consistent compositions.

The **composition process "Robust Ra_4C" is characterized by**:

- $S_{WSC}^{candidate}$ (the same as returned by Ra_4C);
- $\mathcal{B}_\pi$ and its cardinal $\#\mathcal{B}_\pi$. This represents the number of *Extra Description* that Ra_4C requires to compute robust compositions.

The ***Optimization process* of the composition is characterized** by the following parameters:

- the number of abstract semantic links involves in the composition;
- the number of potential candidate links that can achieve an abstract link;
- $\#S_{Ws}^{r}$ i.e., the number of Web services involved in the optimal semantic composition;

The Table 7.4 sketches the main results concerning the computation time performances of our reference architecture on the three scenarios in use.

The whole execution of the reference architecture takes from 0.4 to 1.1 seconds depending on the different scenarios in use.

> In the *semantic link based web service composition* based approach, the SLM construction coupled with the semantic reasoning process are the most time-expensive phases.

For instance it takes 1.1 seconds (i.e., 94.4 % of the overall process) to parse and organize 22 Web services in a 10 matrix whereas the Ra_4C process takes less than 10 ms (i.e., less than 1 % of the overall process) to compute robust compositions of Web services. We obtain such results since the SLM construction is also responsible of high interacting with the *Semantic Reasoning Component* to classify and to order semantic links.

Fortunately we remark that SLMs of our scenarios are relatively sparse i.e., approximately 80% of entries are empty. That is why the Ra_4C process is quite fast in the presented scenarios.

The robustness computation does not really perturb the composition process. This sounds correct since robustness is checked only for non robust semantic links.

Area	Main Parameters & Processes	Description of Main Parameters & Processes		Application Domain		
				Telecom $\#\mathcal{T} := 305$	**E-Tourism** $\#\mathcal{T} := 60$	**E-HealthCare** $\#\mathcal{T} := 105$
Composition	**SLM Parameters**	**SLM Construction + Semantic Reasoning (ms)**		**412.1**	**1080.2**	**290.5**
		File Size (bytes)		2865	3190	1304
		#Rows		7	10	5
		#Columns		8	11	6
		Filling Rate F_r (%)		23.2%	14.5%	20%
		$\overline{m}_{i,j}$		1.9	1.1	1.16
	Ra_4C	**Ra_4C (ms)**		**4.8**	**9.4**	**1.8**
		Candidate compositions ($\#S_{WSC}^{candidate}$)		9	2	2
	Ra_4C **+ Robustness**	**Ra_4C + Robustness (ms)**		**8.8**	**10.4**	**2.3**
		$\min \#\mathcal{B}_\pi$		1	0	5
Optimization	**Composition Optimization**	**Optimization (ms)**		**<1**	**<1**	**<1**
		# Abstract Semantic Links		6	4	5
		# Candidate Semantic Links (Average)		1.5	1.25	1.2
		$\#S_{Ws}^{r}$	(from Ra_4C)	8	5	7
			(from Ra_4C+Robustness)	6	5	5
End to End Composition		**Composition + Discovery Process (ms)**		**465**	**1143.8**	**321.8**

Table 7.4: R_a4C based Web Service Composition Tested on Three Scenarios in Use. Legend: ✗ = not supported, ✓= fully supported.

> The *Optimization* process takes a negligible time i.e., below one millisecond in two thirds of scenarios.

The computation time performance of this component is not very high since the number of candidate Web services for optimization is at worst 9 i.e., no complex problem of optimization.

Semantic Link and Causal Law based Web Service Composition

Here, we specially focus on compositions wherein semantic links and causal laws are involved. Web services are then further described by means of preconditions, effects axioms. Therefore we investigate on the computation time of the following processes:

- the processes in charge of *Semantic reasoning*[5]:
  - the *Knowledge Base Loading*;
  - the *DL Reasoning* process.
- the $s_{sl}Golog$ based `FLC` axiomatization (Section 4.2);
- the *Backward Chaining* approach (Section 4.2);
- the *Optimization Composition* process (Chapter 5).

[5]The pre-processing steps *Knowledge Base Loading* and *DL Reasoning* are in charge of inferring semantic link axioms between functional input and output parameters in this approach.

In addition to the evaluation of the latter components, main intermediate results are valued and given during the composition process of each considered scenario. To this end, we study the same parameters and intermediate results as the previous section i.e., $\#S_{WSC}^{candidate}$, $\#\mathcal{B}_{\pi}$, the number of abstract semantic links, the number of potential candidate links, and $\#S_{Ws}^{r}$.

Moreover we will also focus on **the axiomatization of** $s_{sl}Golog$ which is characterized by

- #Axioms i.e., the number of axioms defined in $s_{sl}Golog$. In the three scenarios we have the following rates of axioms: initial situation (20%), causal laws (40%) relating Web services, and semantic link (40%) axioms.
- and its computation time in milliseconds.

The Table 7.5 sketches the main results concerning the computation performances of our approach on the three scenarios in use.

Area	Main Parameters & Processes	Description of Main Parameters & Processes	Application Domain		
			Telecom $\#\mathcal{T} := 305$	**E-Tourism** $\#\mathcal{T} := 60$	**E-HealthCare** $\#\mathcal{T} := 105$
Reason.	**Knowledge Base Loading**	**Knowledge Base Loading (ms)**	**205.1**	**345.6**	**76.4**
	DL Reasoning	**DL Reasoning (ms)**	**95.5**	**220.5**	**40.5**
Comp.	$s_{sl}Golog$	$s_{sl}Golog$ **Axiomatization (ms)**	**55.4**	**150.8**	**41.1**
		#Axioms	69	122	42
	Backward Chaining	***Computation Time* (ms)**	**230.6**	**550.4**	**210.1**
		Candidate compositions ($\#S_{WSC}^{candidate}$)	5	1	1
Optimization	**Composition**	**Optimization (ms)**	**<1**	**0**	**0**
		# Abstract Semantic Links	6	4	5
	Optimization	# Candidate Semantic Links (Average)	1.5	1.25	1.2
		$\#S_{Ws}^{r}$ (from Ra_4C)	8	5	7
End to End Composition		**Composition + Discovery Process (ms)**	**634.7**	**1321.5**	**397.6**

Table 7.5: $s_{sl}Golog$ based Web Service Composition Tested on Three Scenarios in Use.

The conditional planner coupled with s_{sl}Golog computes compositions of Web services wherein all their services are valid and possible. This is one of the main advantages of this approach.

The whole execution of the reference architecture takes from 0.6 to 1.3 seconds depending on the scenario in use. Not surprisingly, this is less faster than the Ra_4C based end to end composition since s_{sl}Golog considers a more expressive level of Web service description. Moreover semantic links together with causal laws are both used as composability criteria.

> In the *semantic link and causal law based web service composition* based approach, the *Backward Chaining*, is the most time-expensive phases.

Indeed the *Backward Chaining* computation time is more than 30 % in general. The process related to *Knowledge Base Loading* and *DL Reasoning* takes less than 50 % of the process which is quite important. For instance it takes 0.55 second (i.e., 41.6 % of the overall process) to compute the composition result of a problem with 22 Web services in $s_{sl}Golog$.

On contrary the axiomatization step is one of the less time-expensive phases (i.e., less than 12 % in general).

In the same way as the previous Section, the *Optimization* process takes a negligible time i.e., below one millisecond in two thirds of scenarios.

The computation time performance of this component is not very high due to the few number of candidate compositions.

7.2.3 Synthesis

This first set of experiments has been designed to evaluate an end to end composition with its main components (i.e., *Service Discovery and Selection*, *FLC* and *Composition Optimization*) in real scenarios. More specially two different composition components have been tested to achieve *FLC*, mainly depending on the level of Web service description and their composability criteria. The performance of both composition approach is still very good for a once-for-all composition task: to compare, a BPEL4WS programmer would take some hours of work to discover Web services, to ensure semantic links (and causal laws), and (especially) to manually build the BPEL4WS orchestrator.

Not surprisingly, retrieving Web service composition with causal laws and semantic links is more restrictive than computing compositions with only semantic links.

Indeed the Ra_4C based composition approach returns more candidate compositions $S_{WSC}^{candidate}$ than the s_{sl}Golog based composition approach.

The *semantic link and causal law based web service composition* is then more appropriate to compose Web services an higher level of description and then select finer grained compositions of Web services (i.e., services which are both valid and possible).

Even if this approach is more time consuming it also reduces the set of potential candidate compositions by ensuring that composition satisfying causal laws and semantic links. In this direction the approach reduces the number of compositions required by the optimization step and hence highly reduces the computation time of this step.

Applying the Ra_4C based approach on Web service with a high level of description leads to candidate Web service composition with violated causal laws (e.g., open preconditions).

Indeed this approach can compute composition with open preconditions, and cannot satisfy causal laws between services. Two thirds of candidate compositions do not satisfy causal laws. This is the main drawback of Ra_4C.

7.3 Exposition of Our Composition Approach

In this section we conducted a second step of experiments using the implemented prototype system (depicted in Figure 7.1) to focus on the computation time performance and scalability of

the *Functional Level Composition* and *Composition Optimization* components. In other words we turn our attention to different large scale scenarios, which have been randomly generated.

As observed in the previous section and Chapters 4 and 5, the performance of the main reference architecture's components are related to different criteria, for instance

- the SLM size in the Ra_4C based composition;
- the ratio of Knowledge Base Loading, DL Reasoning, $s_{sl}Golog$ Axiomatization, Backward Chaining in the s_{sl} based composition;
- the number of abstract and candidate semantic links in the *Optimization* process;

Here we suggest to study in details the (positive or negative) impact of these parameters on the main reference architecture's components.

The system configuration used for such experiments is the same as the first step of experiments i.e., Intel(R) Core(TM)2 CPU, 1.86GHz with 512 RAM. The PC runs Linux-gnu (2.6.12-12mdk) and Java 2 Edition v1.5.0_11.

7.3.1 The Ra_4C based Composition Performance

In the following we suggest to study which parameters have the most (positive and negative) impact on the computation time performance of the Ra_4C based Composition.

Context of Evaluation

In this experiment SLMs together with their service goal $s_g := \langle \mathcal{A}, \beta \rangle$ have been randomly generated. We assume, without loss of generality, that the generated service goals are not trivial i.e., β cannot be directly satisfied by the initial conditions $\mathcal{A}$. In other words the goal β can only be satisfied by a composition of at least one semantic Web service.

Given this context, we will draw some hard dependences between the computation time performance of the Ra_4C process and the following three parameters:

- the parameters of the service goal s_g i.e., its number of instance in $\mathcal{A}$ and its number of goal in β.
- the main features of SLMs i.e., their #Rows, #Columns, the number of relevant Web service $\#S_{Ws}$ (and implicitly the number of parameters required to describe Web services) define in the SLM;
- the filling rate of the SLM F_r and the average of elements by non empty entry $\overline{m}_{i,j}$ in this same SLM.

Results

According to the practical experiments we first draw some trivial but important results concerning the properties of SLMs:

- **the more Web services in S_{Ws} the higher the filling rates of SLMs with a fixed size of rows and columns** (Figure 7.6). Technically, we fixed the number of rows #Rows of the SLM, $\#\mathcal{A}$ and $\#\beta$ and computed the filling rates F_r depending on the number of relevant Web services $\#SWs$ (and their number of functional parameters).

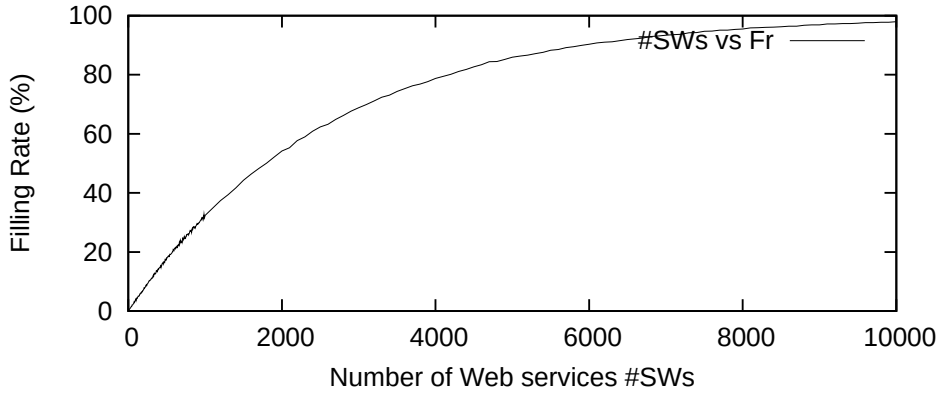


Figure 7.6: Computation of the Filling Rate F_r of SLMs with 100 Rows by Varying the Number of Web Service (with 4 $\#InputMax$ and 4 $\#OutputMax$) in $\#S_{Ws}$.

- **the more Web services in S_{Ws} the higher the average of elements by non empty entry $\overline{m}_{i,j}$ in the SLM**. This dependence is linear. Technically, we fixed the number of rows #Rows of the SLM, $\#\mathcal{A}$ and $\#\beta$ and computed $\overline{m}_{i,j}$ depending on the number of relevant Web services $\#SWs$ (and their number of functional parameters).

We can conclude by interpreting these first trivial results that **the less Web services (and its number of functional parameters) involved in the SLM the less complex and the sparser the SLM**. Therefore the *Service Discovery and Selection* has a key role to reduce the complexity of the SLM.

By running our composition approach on random SLMs and service goal, we also obtain the following more subtle and interesting results:

- **the more expressive is the initial conditions $\mathcal{A}$ of the AI planning-based Web service composition $\langle S_{Ws}, \mathcal{A}, \beta \rangle$ the faster is the composition process (Figure 7.7)**. Technically, we fixed the number of rows #Rows of the SLM, $\#\beta$, the number of relevant Web services $\#SWs$ (and their number of functional parameters) and evaluated the computation time performance of the Ra_4C process, depending on $\#\mathcal{A}$.

- **the sparser the SLM the faster is the Ra_4C process** (Figure 7.8 and 7.9). Technically, we fixed the number of rows #Rows of the SLM, $\#\mathcal{A}$, $\#\beta$ and computed the computation time performance of Ra_4C, depending on the filling rates F_r to obtain results in Figure 7.8.

  The results of Figure 7.9 has been obtained by fixing $\#\mathcal{A}$, $\#\beta$ and evaluating the computation time performance of the Ra_4C process, depending on the filling rates F_r and the number of rows #Rows of the SLM.

- **the less number of functional parameters by Web services the faster is the Ra_4C process** (Figure 7.10). Technically, we fixed the number of rows #Rows of the SLM, $\#\mathcal{A}$, $\#\beta$ and evaluated the computation time performance of Ra_4C depending on the number of functional parameters by Web services to obtain such results.

According to the previous results, the SLM formalism coupled with the Ra_4C composition approach scales very well in case i) SLMs are *relatively* sparse (i.e., a filling rate $F_r \leq 40\%$) and ii) the initial condition $\mathcal{A}$ of the composition problem is expressive enough (i.e., $\#\mathcal{A} > \frac{1}{3}\#Rows$).

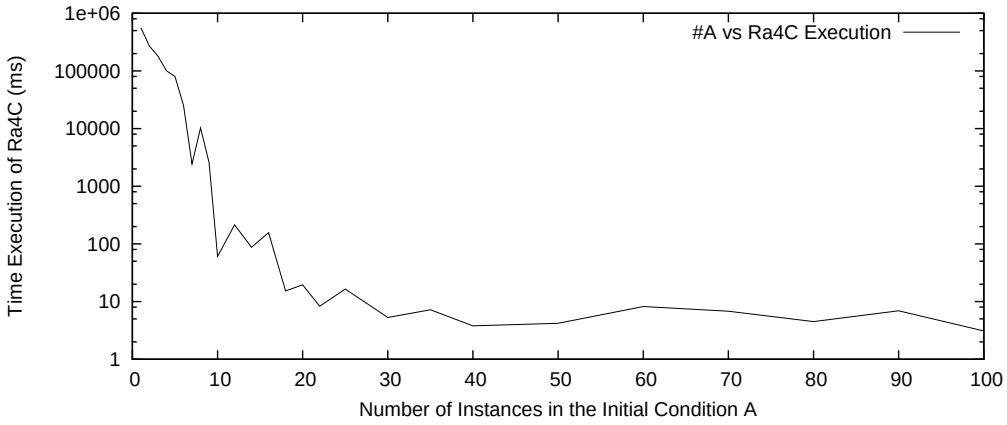


Figure 7.7: Computation Time of the Ra_4C based Composition Approach with an SLM of $\#Rows = 100$ and $F_r = 15\%$ by Varying the Number of Instances in $\mathcal{A}$.

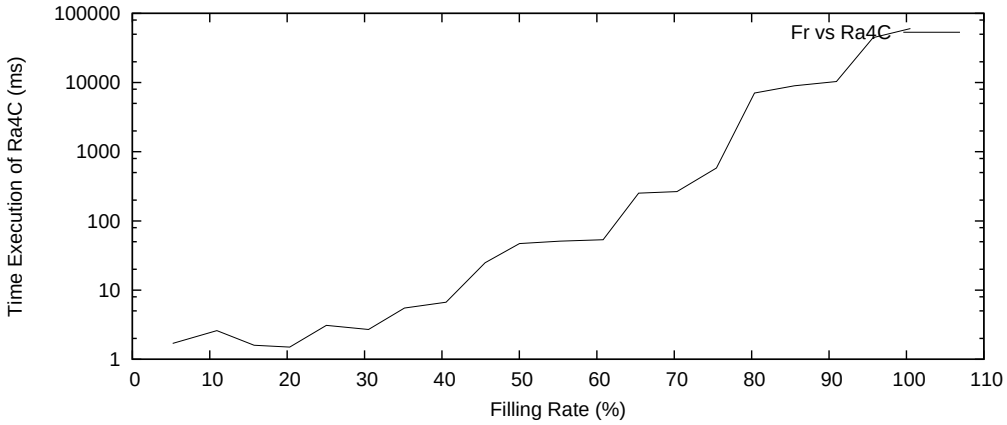


Figure 7.8: Computation Time of the Ra_4C based Composition Approach with $\#\mathcal{A} = 30$ and an SLM of $\#Rows = 100$, $\overline{m}_{i,j} = 1$ by Varying the Filling Rate F_r of the SLM.

Most of scenarios (i.e., scenarios presented in these Ph.D studies and others confidential) we study in this Ph.D thesis meet these constraints. For example, as observed in Table 7.4, the three scenario in use are in such a configuration. The *Telecommunication* scenario considers the composition problem $\langle S_{Ws}, \mathcal{A}, \beta \rangle$ wherein $\#\mathcal{A}$ is 3, the number of row of the SLM is 7, and its filling rate F_r is 23.2%.

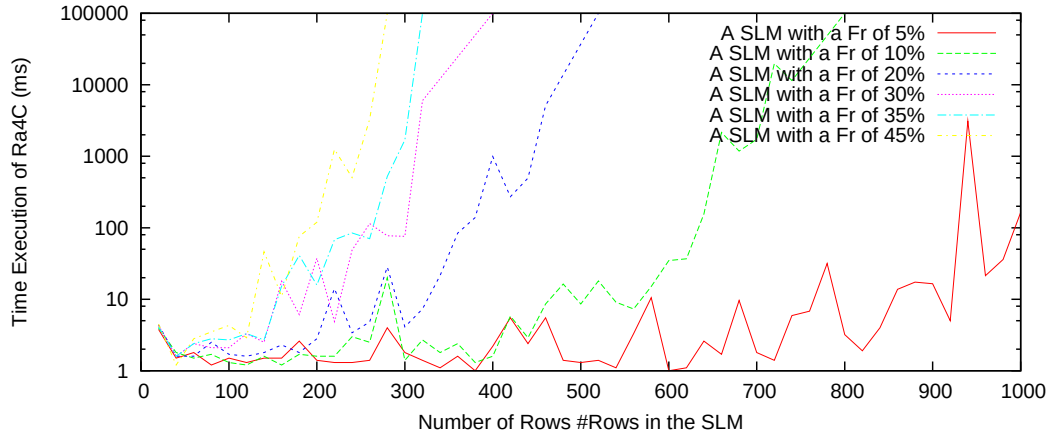


Figure 7.9: Computation Time of the Ra_4C based Composition Approach with $\#\mathcal{A} = \frac{\#Rows}{3}$ and an SLM Defined by $\overline{m}_{i,j} = 1$ by Varying the of the Number of Rows in the SLM.

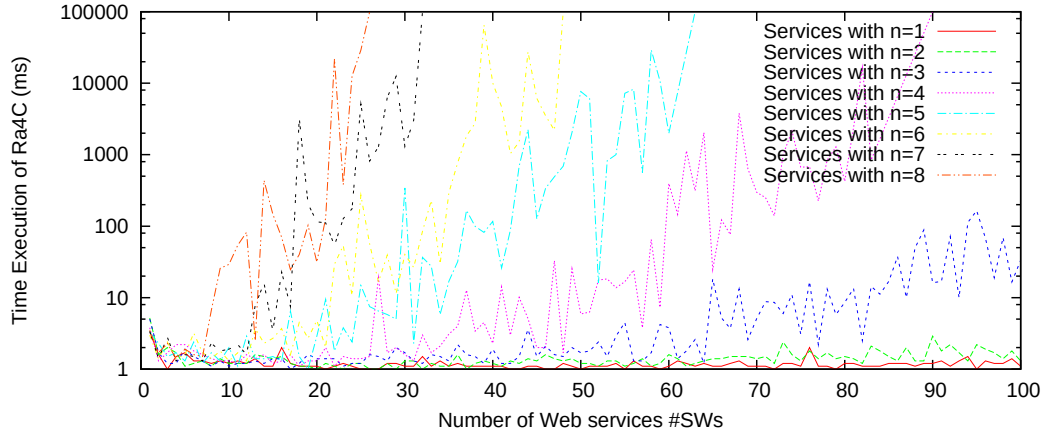


Figure 7.10: Computation Time of the Ra_4C based Composition Approach with an SLM of $\#Rows = 100$, $\#\mathcal{A} = 30$ by Varying the Number of Web Services $\#S_{Ws}$ with 'n' their Maximum Number of Input and Output Parameters.

Some Concluding Remarks on Performance of Ra_4C

As previously observed in Chapter 4 and confirmed in experimental results (Table 7.4):

The performance of the Ra_4C based composition is mainly depending on

i) the number of instances defined in the initial situation of the composition problem;

ii) the size of the target SLM and then its filling rate;

iii) the number of Web services and their number of functional parameters.

The SLM is a very interesting feature to perform Web service composition not only with good properties (correctness, completeness, robustness, consistency) but also with good performance since **the SLM model is responsible of retrieving and pre-compile all relevant semantic links (at design time) we could encounter in a composition of services**.

Once the SLM model is computed the composition process speed-up since semantic reasoning has been first achieved in a pre-processing step.

Moreover such a matrix can be re-used without major changes for close composition problem. Therefore this ensures the flexibility of this approach.

As studied in Chapter 4.1,

The main limitation is related to the levels of Web service description and composability criteria (here restricted to semantic links).

The efficiency of the composition process Ra_4C is related to the performance of the discovery process i.e., the set of discovered Web services.

Indeed the less Web services involved in the set of relevant Web services S_{Ws} (and in the SLM), the smaller the size of the SLMs and faster the process of composition Ra_4C. In this experimental result we study Web service composition by applying a very naive algorithm of Web service discovery. One way to easily improve the computation-time performance of the process Ra_4C consists first in applying a more efficient *Service Discovery and Selection* process e.g., [24].

7.3.2 The $s_{sl}Golog$ based Composition Performance

In the following we establish if our approach of conditional planning by using semantic links together with causal laws to control search and establish composition is efficient. To this end we study in details the computation time performance and scalability of the main processes required by the $s_{sl}Golog$ based composition approach i.e.,

- the processes in charge of *Semantic reasoning*:
  - the *Knowledge Base Loading process*;
  - the *DL Reasoning*.
- the $s_{sl}Golog$ *Axiomatization*;

- the *Backward Chaining* approach (we modified to support s_{sl}Golog).

In addition we compare the computation time performance of the overall composition process with the pure *sGolog*, first introduced by [103] to compose Web service.

Context of Evaluation

In this experiment, parameters of services are referred as DL concepts in an $\mathcal{ALE}$ Ontology (305 concepts and 117 properties). This experimentation consists of two different classes of scenarios:

- Ten scenarios $\omega^{A}_{i,1\leq i\leq 10}$ wherein $i\times 7$ services are involved. Each service is described with 2 input, 2 output parameters and 4 precondition and 4 effect axioms. Moreover we have the following rates of axioms: initial situation (20%), causal laws (40%) relating Web services, and semantic link (40%) axioms. The number of axioms is defined by $i\times 7\times 5$. Here we simply evaluate performance of each component of the composition approach;
- Ten scenarios $\omega^{B}_{i,1\leq i\leq 10}$ wherein 49 services described with i input, i output parameter(s) and 4 preconditions, effects axioms. The number of initial situation, causal laws relating Web services, and semantic link axioms is $i\times 49\times 5$. However, here, we changed the rate of its axioms. The rate of initial situation is still 20%, whereas the rate of causal laws axioms is $(i\times 49\times 5)-(\frac{2}{100}\times i\times 49\times 5)$ and the rate of semantic link axioms is $(i\times 49\times 5)+(\frac{2}{100}\times i\times 49\times 5)$. Here we evaluate impact of semantic links axioms on the composition process.

Semantic links and causal laws are then both involved and used to compute compositions of Web services.

In both classes of scenarios the main steps of the composition process is as follows. The *Knowledge Base* of the ontology (actually its Terminological and Assertional Box) is loaded, and then DL reasoning, s_{sl}Golog axiomatization and conditional planning through *Backward Chaining* take place. In both classes of scenarios the *DL reasoning* is done at design time, all semantic matches are computed before running the conditional planner. It is straightforward to apply *DL reasoning* on demand e.g., retrieving the match type of a semantic link at run time.

Results

According to the practical experiments we obtain the following results concerning the *sGolog* based approach.

- not surprisingly, **the *Backward Chaining* process is more time consuming than the three other processes of the s_{sl}*Golog* based composition** (Figure 7.11 and 7.12). In general *Knowledge Base Loading*, *DL Reasoning* and s_{sl}*Golog* Axiomatization are still linear with the number of Web services and their input and output parameters descriptions;
- the **DL reasoning has direct positive impact on the planning performances** (Figure 7.11 and 7.12). Our approach still outperforms the *sGolog* based approach. Indeed s_{sl}*Golog* i.e., sGolog coupled with information on semantic links scales better than sGolog (without DL reasoning on semantic links). Indeed the composition approach is directed by semantic links, which rapidly prunes the search space in a composition problem. Therefore **the planning task still scales well if semantic links are known at composition time**;
- **the more Web services, the more computation time of *DL reasoning* in comparison to the *Knowldege Loading* and *Axiomatization*** (Figure 7.11);

- in the same way as the Ra_4C based approach, **the more Web services, the less the computation time performance** (Figure 7.13). Indeed the rate of semantic links and causal laws required in this case increases with the number of Web services.

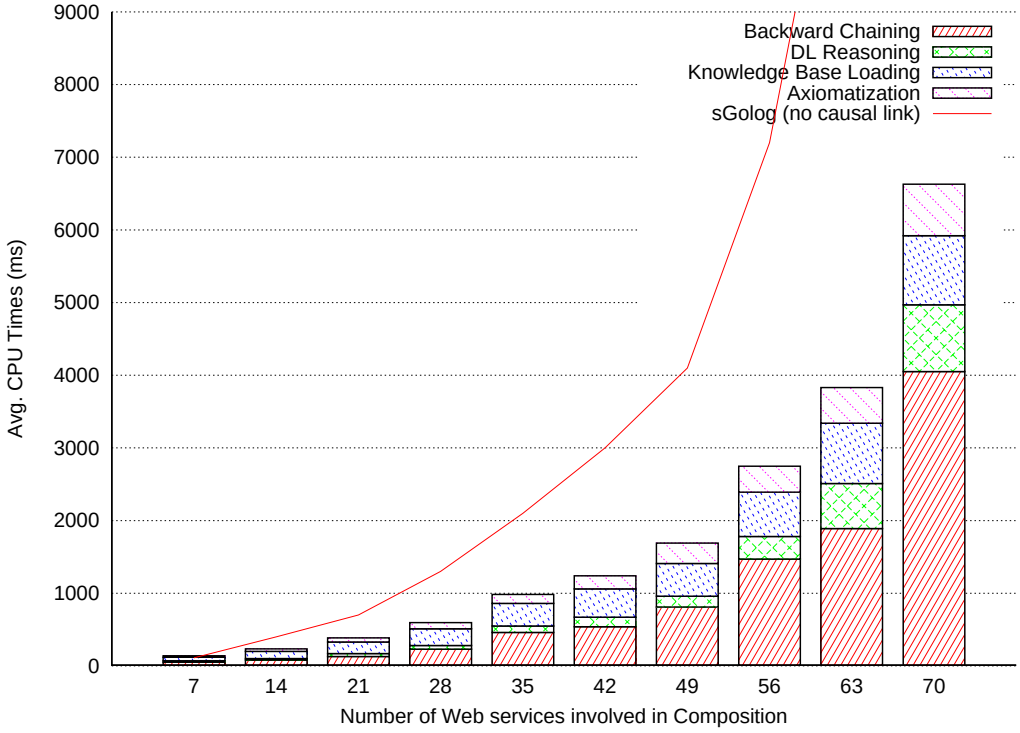


Figure 7.11: Knowledge Base Loading, DL Reasoning, Axiomatization and Planning Processes on Scenario ω^A.

- in the same way as the Ra_4C based composition, **the more input and output parameters by Web services and semantic links axioms the worst the computation time performance** (Figure 7.12). In scenario ω_B, *DL reasoning* requires more time than in scenario ω_A since more *DL reasoning* are computed e.g., for 49 services and 9 input and 9 output parameters, $(9 \times 49)^2$ subsumption tests are performed in the worst case.

As previously remarked, the Ra_4C based approach does not consider causal laws, and so scales very well.

- It is obvious that **composing Web service with semantic link and causal laws is more computation time consuming than composing Web services with only its semantic links** (Figure 7.13). However many composition scenarios cannot be performed by the Ra_4C based approach i.e., scenarios that consider preconditions, effects on Web services, and semantic links, causal laws involved in the composition. It is obvious that considering causal laws in a composition is the hard task to achieve.

According to the previous results, the $s_{sl}Golog$ based approach scales well in case i) the number of Web services is less than 70, ii) the number of input and output parameters is 2,

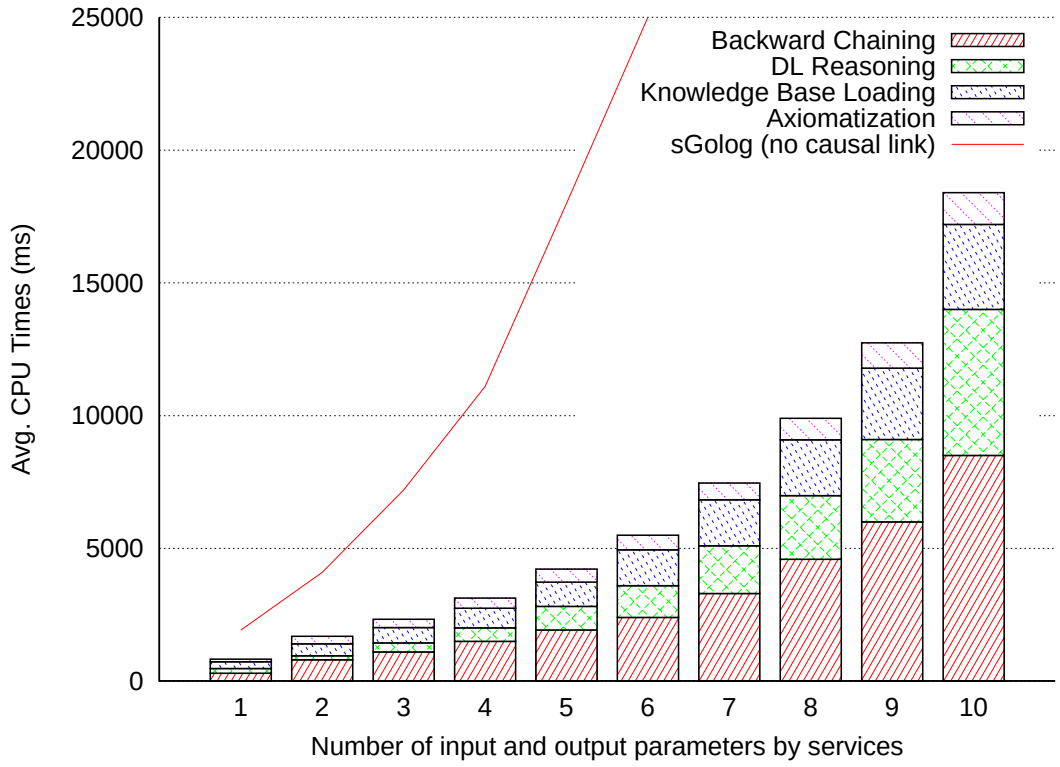


Figure 7.12: Knowledge Base Loading, DL Reasoning, Axiomatization and Planning Processes on Scenario ω^B.

iii) the number of precondition and effect axioms is 4, iv) the number of initial situation (20%), causal laws (40%) relating Web services, and semantic link axioms (40%) is 350. Unfortunately we did not focus on more complex scenarios. However, this seems an interesting trade-off to achieve composition with semantic links and causal laws.

As observed in Table 7.5, the three scenarios in use work with a more restrictive configuration.

Some Concluding Remarks on Performance of s_{sl}Golog based Approach

Even if using conditional plans includes sometimes unrealistically high number of alternative paths, the DL reasoning based semantic link and causal laws axioms are used i) to reduce the search space ii) to reduce the number of these alternative plans by retrieving more relevant composition of Web services, iii) to improve performance of the composition process.

As previously observed in Section Table 7.5 and confirmed in experimental results, the performance of the s_{sl}Golog based composition is mainly depending on

i) *DL Reasoning*;

ii) *Backward Chaining*.

Indeed the Knowledge Base Loading and s_{sl}*Golog* Axiomatization processes are much less time computation consuming.

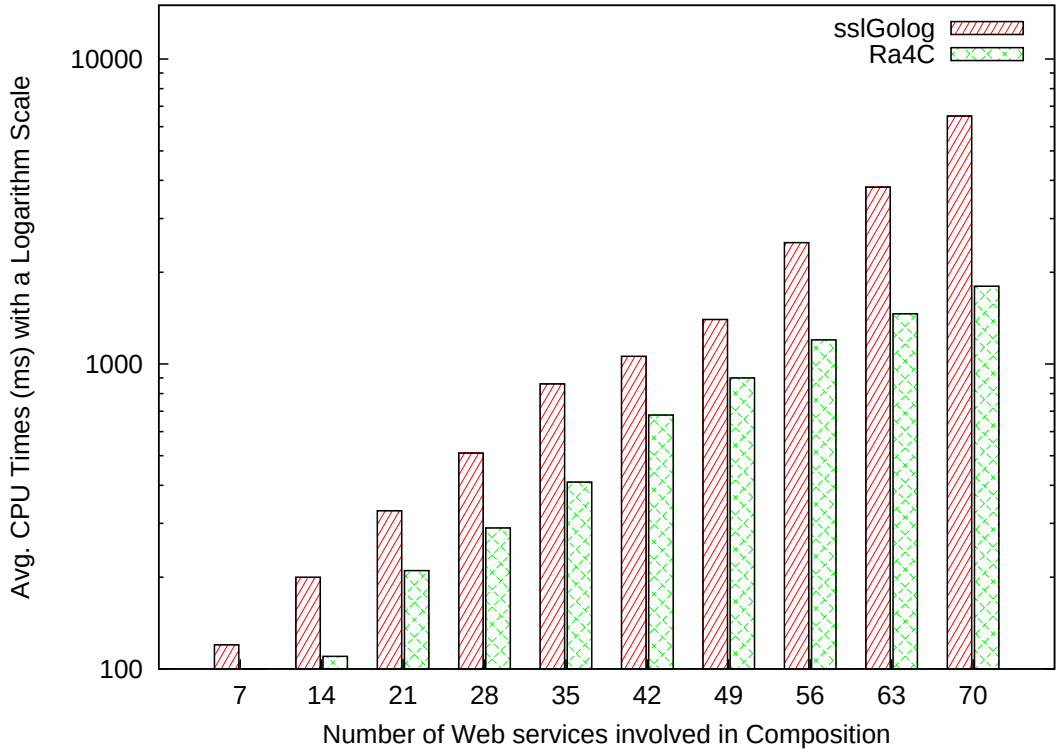


Figure 7.13: Computation-Time Performance of s_{sl}Golog and Ra_4C on Scenario ω^A.

The s_{sl}Golog based approach is a very interesting approach to perform Web service composition wherein input, output parameters, preconditions, effects, semantic links and causal laws are required to achieve the composition process.

There is a trade-off between semantic links and causal laws expressivity since

- **the more input and output parameters, the more semantic links and the more the computation time of DL reasoning** (in Figure 7.12, the available number of semantic links and causal laws axioms are the same but not their numbers of useful and used axioms);
- **the more causal laws the more the computation time of the Backward Chaining approach**;

A limitation of this composition approach is related to the overall computation time. Since this approach is mainly depending on DL reasoning,

The main limitation is related to the number of Web services' parameters and expressivity of ontology we use to describe services.

In case Knowledge Base loading and DL reasoning on functional parameters has been first pre-compiled (for instance in an SLM), the computation time of the overall composition process is highly reduced.

In further experiments, we plan to investigate a trade-off between DL expressivity and AI planning for Web service composition.

7.3.3 The *Optimization* Process Performance

In the following we suggest to study the scalability of the *Optimization Composition* approach (i.e., the global selection based approach by Integer Linear Programming). In addition this experiment aimed at comparing the three selection approaches previously described in Chapter 5 i.e.,

- *local optimization based approach*;
- *global selection by exhaustive search*;
- *global selection by integer programming.*

with respect to the computational overhead involved by their selection phase. The idea is to provide a basis for determining when should global selection be preferred over local optimization. Accordingly, we measured the computation time performance (in milliseconds) of selecting candidate semantic links to create practical compositions under the three different selection approaches. For each test case, we executed the process 10 times and computed the average computation cost.

Finally this experiment also evaluates and compares the semantic quality of different practical composition of Web services computed by *local optimization* and *global optimization.*

Context of Evaluation

In our experiments we assumed that robustness, common description rate and matching quality of each semantic link have been inferred in a pre-processing step of semantic reasoning and calculated using the formulas presented in Section 5.2.

On the one hand, the global selection process is invoked only once during a composite service execution. In the case of local optimization, on the other hand, if we assume that the number of abstract links that are executed is n, then the semantic links selector is invoked n times to select a candidate semantic link for each abstract link.

The experiments involved composite Web services with varying the numbers of abstract and candidate semantic links. The composite services were created by randomly defining abstract and candidate links in the composite service. The number of abstract links varied from 100 to 500 with steps of 25 (e.g., 100, 125, ... 500). Also, we varied the number of candidate semantic links per abstract link from 10 to 150 with steps of 10.

Results

Figures 7.14 and 7.15 plot computation time (in milliseconds) of selecting semantic links for Web service composition. In this experiment, we varied the number of abstract links and the number of candidate links per abstract link. In the three approaches, we observe that:

- **the computation time increases when the number of abstract links increases and the number of candidate links increases**.

The computation time of global selection by exhaustive search is very high even in very small scale in aspect of the number of abstract semantic links and their candidate links. Although **the computation time cost of global selection by IP is higher than that of local optimization**, it still acceptable if the number of abstract links and candidate links is not very large. For example, finding the optimal solution to the optimization problem takes 10 seconds for a composition of 450 abstract semantic links with 100 candidate links (i.e., 10 candidate services by task). This is almost 4 times higher than the local optimization approach (2.65 seconds).

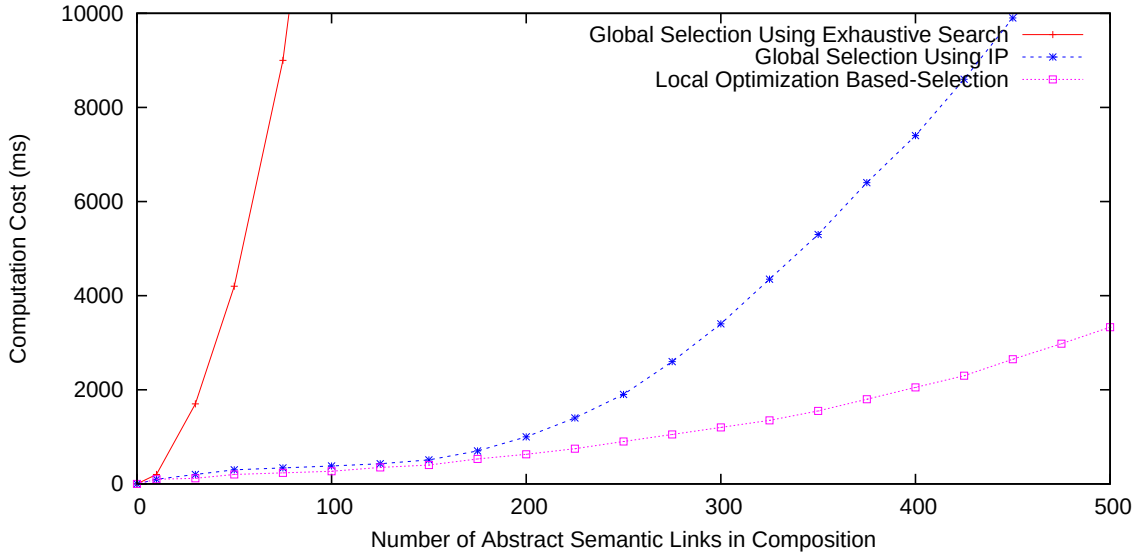


Figure 7.14: Computation Time for Optimal Practical Composition by Varying the Number of Abstract Semantic Links (with 100 candidates for each abstract links).

Tables 7.6 and 7.7 present experiment results on composites'*robustness* (i.e., Q_r in $[0, 1]$), *common description rate* (i.e., Q_{cd} in $(0, 1]$) and *matching quality* (i.e., Q_m in $(0, 1]$) of practical composition of Web services, where we vary the number of abstract semantic links.

The experiment results show that the global selection approach yields significantly better semantic quality than the local optimization approach. For example, the *common description rate* is consistently higher when using global planning approach than the *common description rate* of the local optimization approach.

Some Concluding Remarks on Performance of the Optimization Composition Approach

The optimization problem formulated in Chapter 5, which is equivalent to an Integer linear programming problem, is NP-hard [155]. In case the number of abstract and candidate semantic links in the system is expected to be very high (of the order of millions or higher), finding the exact optimal solution to such a problem takes exponential run-time complexity in the worst case, so no practical.

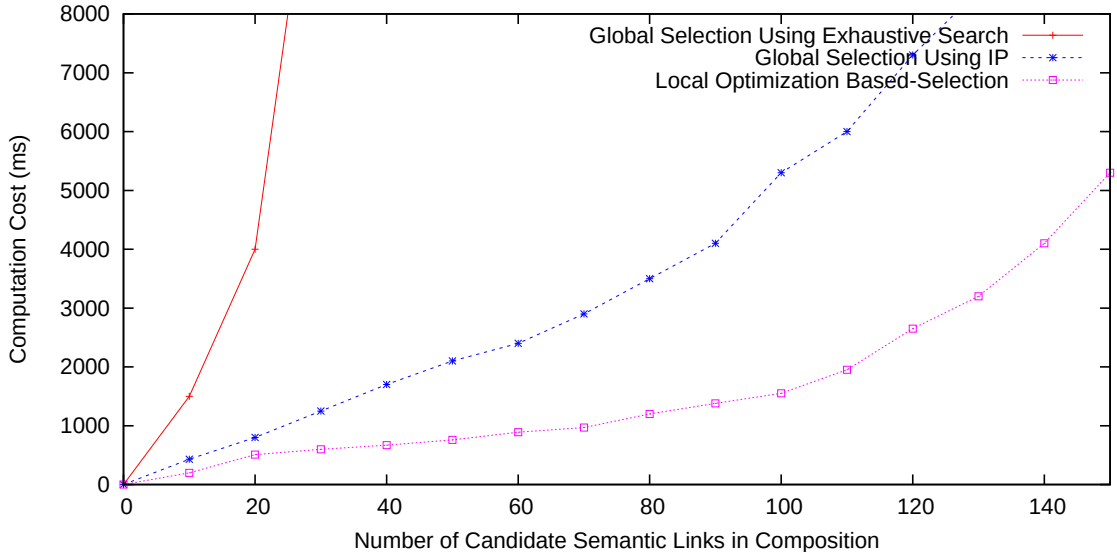


Figure 7.15: Computation Cost for Optimal Practical Composition by Varying the Number of Candidate Semantic Links (with 350 abstract semantic links).

Composite Web Service c	$Q_r(c)$		$Q_{cd}(c)$		$Q_m(c)$	
	Global	Local	Global	Local	Global ($\times 10^{-7}$)	Local ($\times 10^{-7}$)
1	0	0	0.85	0.71	8.12	0.31
2	0	0	0.67	0.61	0.53	0.12
3	0	0	0.50	0.3	9.44	9.34
4	0	0	0.78	0.6	23.53	2.5
5	0	0	0.81	0.76	100.44	19.4
6	0	0	0.77	0.77	0.01	0.01
7	0	0	0.71	0.56	1.1	1.01
8	0	0	0.80	0.78	12.1	0.91
9	0	0	0.67	0.6	0.08	0.03
10	0	0	0.77	0.5	0.12	0.02

Table 7.6: Experimental Results on Semantic Quality of Composite Services (Composite Web services have 20 Abstract Semantic Links).

Our approach scales well by running a heuristic based IP solver wherein hundreds of abstract and candidate semantic links are involved. This is a suitable upper bound for practicable industrial applications.

From the experimental results, we conclude that the IP based global selection approach leads to significantly better semantic quality of Web service composition with little extra computation cost. For example, a composite service with 250 abstract links and 100 candidate links per abstract link spends: 1) 1.9 seconds for selecting Web services using global selection; 2) 0.9

Composite Web Service c	$Q_r(c)$		$Q_{cd}(c)$		$Q_m(c)$	
	Global	Local	Global	Local	Global ($\times 10^{-16}$)	Local ($\times 10^{-16}$)
1	0	0	0.52	0.49	81.2	3.11
2	0	0	0.61	0.58	4.31	0.34
3	0	0	0.98	0.7	1.34	1.10
4	0	0	0.39	0.28	3.31	0.78
5	0	0	0.78	0.7	1.45	0.98
6	0	0	0.31	0.28	1.12	0.01
7	0	0	0.68	0.60	101.33	14.65
8	0	0	0.45	0.39	45.33	44.55
9	0	0	0.90	0.78	8.54	7.56
10	0	0	0.70	0.60	34.11	31.67

Table 7.7: Experimental Results on Semantic Quality of Composite Services (Composite Web services have 50 Abstract Semantic Links).

seconds using local optimization.

These results reinforce the conclusions of the analytical considerations of Chapter 5.

> If there is no requirement for specifying global constraints, then local optimization is preferable.

Global planning is superior when it comes to selecting semantic links that satisfy certain global constraints and which optimize global tradeoffs. Given the fact that, a global selection approach considers both local and global selection constraints, the results clearly demonstrate the benefit of using a IP-based method for global semantic links selection, even if we take into consideration the modest selection overhead.

> The evaluation and theoretical results showed that our global selection based approach is not only more suitable than the local approaches, but also outperforms the naive approach.

As highlight in Chapter 5, in case of higher number of abstract and candidate links, the problem can be, for instance, divided in several *global selection problems*. Alternatively, suboptimal solutions satisfying revisited quality thresholds can be sufficient.

7.3.4 Synthesis

In this section we conducted a second step of experimentation using the implemented prototype system (depicted in Figure 7.1) to evaluate the computation time performance and scalability of components of the reference architecture presented in Chapters 4 and 5 i.e., the two *Functional Level Composition* based approaches and the *Composition Optimization* method.

From this experimentation we obtained valuable experimental results on the latter components. For instance we retrieved parameters that have a direct and important impact of the Ra_4C based composition approach. Moreover we study computation time performance of Web

service compositions that require semantic links and causal laws as composability criteria. Finally we established a trade-off between the number of abstract semantic links and the number of candidate links for the optimization based composition method.

According to these experimentations, we can draw the *best practices* for Web service composition in Table 7.8.

Process	Parameters	Computation Time in ms			
		(0, 1000]	(1000, 2000]	(2000, 5000]	(5000, 10000]
Semantic Links based Web Service Composition	Nb services	69	74	78	83
	Nb Inputs, Outputs	4	4	4	4
Semantic Links and Causal Laws based Web Service Composition	Nb services	35	53	65	71
	Nb Inputs, Outputs	2	2	2	2
	Nb Preconditions, Effects Axioms	4	4	4	4
Composition Optimization	Nb Abstract semantic links	220	260	350	450
	Nb Candidate semantic Link	100	100	100	100

Table 7.8: Best Practices for Web Service Composition.

7.4 Conclusion

In this Chapter we discussed the prototype tool that we developed to compute automated semantic Web service compositions and its optimal composition in our framework. In this direction we presented a reference architecture to realize Web service composition (Figure 7.5). This reference architecture support **requirements** $R_{Applicability}$ introduced in Chapter 2.

In addition our Web service composition model have been evaluated on two different experimentations. The first experiments have been directed by three scenarios in use in France Telecom whereas the second experiments study computation time performance and scalability of the innovative components of the reference architecture.

From a general point of view:

Even if our approaches scale well in France Telecom scenarios with its best practices, scalability [51] of Web service composition is still an open issue for very complex and expressive scenarios e.g., large number (e.g., thousands) of Web services, semantic links and their candidates with expressive descriptions.

Both experiments reinforce the conclusions of the analytical considerations of Chapters in Part II by demonstrating the benefit of the different innovative components in different scenarios.

Conclusion and Perspectives

In this chapter, we will summarize our Ph.D work. Then we sketch our main contributions and align them to Chapters of this Ph.D work and to our own publications, in which the results have been published. Furthermore, we will give some ideas that can be of further research interest.

Summary and Contributions

In this thesis we have made several contributions to the field of service-oriented computing and semantic Web services, aiming at resolving an open issue that has emerged in recent couple of years: *Semantic Web service composition wherein services are characterized in terms of their functional properties.*

In more details we described two complementary approaches to automatic Web service composition. Our approaches have been directed to meet the main challenges in service composition. First, both approaches are autonomous so that the users do not require to analyze the huge amount of available services manually. Second, they have good scalability and flexibility so that the composition is better performed in a dynamic environment. Third, they solve the heterogeneous problem because the Semantic Web information is used for matching and composing Web services. The applicable scenario for this approach is: given the specification of available Web services and user's requirement, an automated agent or program can generate a composition of available services which satisfies the requirement of the user. The result generation process is fully autonomous without the intervention from the user. The process generation should rely on the specification of Web services, including its functionalities attributes.

In this Ph.D work, we focused on the following key points:

Analysis of Related Work

First of all we have studied and analyzed related work in the field of automated Web service composition in order to draw main requirements and then to address this issue. Therefore the first part of our Ph.D study focused on these following six meta requirements; namely

- **Automation** ($R^{Composition}_{Automation}$) to achieve automation of composition;
- **Expressivity** ($R_{Expressivity}$) to support expressive descriptions of Web services ($R^{Service}_{Expressivity}$) on the one hand, and expressive descriptions of compositions ($R^{Composition}_{Expressivity}$) on the other hand;

- **Composability** ($R_{Composability}$) criteria pertaining automatic composition of services, described in terms of input, output parameters and preconditions, effects. Such citeria are used to relate some Web services in a composition i.e., $R^{Semantic}_{Composability}$ and $R^{Causal}_{Composability}$;
- **Flexibility** ($R_{Flexibility}$) to ensure adaptability of our composition model;
- **Optimzation** ($R_{Optimization}$) to compute optimal compositions of Web services;
- Industrial **Applicability** ($R_{Applicability}$) to ensure applicability of our approach to standard proposals i.e., $R^{Service}_{Applicability}$ and $R^{Composition}_{Applicability}$.

Part II of this Ph.D thesis addressed all these requirements for the success of Functional Level Composition of Web services.

Semantic Link and SLM

The thesis introduced the semantic link concept between Web services as a composability criterion in Web service composition. This connection (Requirement $R^{Semantic}_{Composability}$), considered as the main issue to form new value-added services by composition at functional level, is required to semantically link output to input parameters of Web services (a part of Requirement $R^{Service}_{Expressivity}$). This is detailed in Section 3.1 of this Ph.D report.

The thesis highlighted the issue related to robustness of semantic links and presents different techniques to solve the problem of robustness in Web service composition. This was detailed in Sections 3.1.4 and 3.1.5 of this Ph.D report.

The thesis laid out a framework SLM (i.e., the Semantic Link Matrix) for pre-computing and capturing all relevant semantic links in a formal and flexible (Requirement $R_{Flexibility}$) model to achieve Web service composition. Moreover such model supports expressive compositions of Web services (Requirement $R^{Composition}_{Expressivity}$). This was detailed in Section 3.2 of this Ph.D report.

Web Service Composition

The thesis fits semantic Web service composition to a semantic links composition by means of the feature of sequence composability between services. Towards this issue we suggested to apply an AI planning-based technique for computing correct, consistent, complete and optionally robust Web service compositions (Ra_4C) on an SLM of a given domain. The result of the latter computing is a partial ordering of Web services arranged in a simpler version of a workflow (Requirement $R^{Composition}_{Expressivity}$) that fulfils a given composition goal. This was detailed in Section 4.1 of this Ph.D report.

The thesis provided a composition approach that considers semantic links together with causal laws as composability criteria. This approach required a more expressive level of Web services description. Indeed functional input and output parameters of Web services have respectively preconditions and effects on the application domain. Towards this issue an augmented and adapted version of the logic programming language Golog i.e., $s_{sl}Golog$ is presented as a natural formalism not only for reasoning about the latter links and laws, but also for automatically composing services. This approach supports Requirements $R^{Service}_{Expressivity}$, $R^{Composition}_{Expressivity}$ and conditional compositions of services. This is detailed in Section 4.2 of this Ph.D report.

Web Service Composition Optimization

The thesis provided a technique for computing semantic link based *optimal* Web service composition (Requirement $R_{Optimization}$). To this end innovative and distinguishing criteria have been introduced to estimate the overall semantic quality of any composition. Therefore non functional criteria such as quality of service (QoS) are no longer considered as the only criteria to rank compositions satisfying the same goal. Roughly speaking, a general and extensible model to evaluate and estimate quality of both elementary and semantic links based Web service composition was presented. From this the semantic link based *optimal* Web service composition was computed by i) a local, ii) a naive and ii) a global selection approach. The latter approach was formulated as an optimization problem which was solved using efficient integer linear programming methods. This was detailed in Chapter 5 of this Ph.D report.

Implementation, Experimentation and Evaluation

The thesis elaborated an architecture wherein Web services can be discovered, selected, and composed at Functional Level. This was detailed in Chapter 7 of this Ph.D report.

Our Ph.D work has an efficient implementation and it provides a promising solution for Web Service composition problems. The thesis presents an open source prototype tool that implements our technique for automatically computing compositions of Web services and rendering them in a standard format (Requirement $R^{Composition}_{Applicability}$). This was detailed in Chapter 7 of this Ph.D report.

The thesis illustrated our approaches in different levels of scenario i.e., from scenario in Use in Telecom, to more experimental scenarios in domains of E-Tourism, E-HealthCare domain, and to random scenario to evaluate scalability of our approach. As highlighted by Chapter 6, the spectrum of applications seems wider.

Concluding Remarks

The empirical evaluation of the system performance presented in Chapter 7 illustrated the experiences made with the application of the proposed techniques and leads to the conclusion that the concepts presented can be applied to practical and industrial scenarios in use e.g., *Dynamic and automated configuration of Web services in the France Telecom's information system.*

In addition we have drawn *best practices for Web Service Composition* in Table 7.8.

On the one hand this confirms that Web service composition can be realized in *Industrial scenarios* and in a semantic context, the whole with basic composition constructs.

On the other hand *composition of thousand of Web services* is *not yet a reality* since the latter hard level of composition requires first solutions to other issues related to *discovery*, *selection*, *interoperation*, *mediation*, *interaction*, *verification*, *monitoring/management* and also *execution*.

Impact of our Contributions

The presented thesis provided conceptual and technical contributions that have been reviewed, approved and published in major conferences (see publications in Table 7.9 aligned with our contributions) of the following *Research areas*:

- *Service Computing*: Web Service, Service Composition, Composition Optimization;
- *Artificial Intelligence*: Semantic Web, Knowledge Representation, Automated Reasoning, AI Planning.

In addition parts of this Ph.D works have been incorporated in several European projects such as the *European Union Project EU-IST-2004-507482 Knowledge Web* and *European Union Project EU-IST-2006-027617 Spice* (see the *Industrial Activities* column of Table 7.9).

Perspectives

The field of semantic Web services composition is a relatively new research area, and there are several interesting research questions that have not been addressed by the present thesis. However, the framework presented in this thesis may be seen as a starting point for future work. In particular, we identified the following research and development activities that may be carried out based on this thesis (ordered from Micro to Macro levels).

A future direction of this work, which is unfortunately out of the scope of the Web services research area, consists in extending our framework with distributed ontologies. Indeed the semantic annotations of Web services do not share the same and unique ontology. The ontology interoperability is a big issue for future success of semantic Web services and their composition.

Extra Properties on Semantic Links

A first interesting issue is related to some inappropriate properties of the DL difference operator we used in our Ph.D work to achieve robustness and to model the quality of elementary semantic links. Indeed this DL difference can be compared to a simple syntactic difference.

More Expressive Web Service Description

One of the most interesting future direction could be to consider more expressive description of Web services. Indeed some standard proposals such as WSMO suggest to reason on the service goal to achieve discovery, selection and composition tasks. This can be helpful to distinguish some identical functional description of Web services, and then to integrate them in a composition process.

More Expressive Composition Constructs

The main direction for continuing the work in Section 4.2 is to combine our work and the approach of [182] to cope first with more large and complex composition, and second with more complex structures like (infinite) loops.

	Ph.D Contribution	Ph.D Report		Academic Publications			European Project Deliverable	International Patent
		Chapter	Section	International				
				Journal	Conference	Workshop		
In the Ph.D Report	Semantic Link Concept	3	3.1	[112]	[114, 115]	[113]	[119]	
	Robustness of Semantic Links	3	3.1.4 3.1.5	[112]	[108, 109]			
	Semantic Link Matrix	3	3.2	[112]	[114, 115]	[113]	[119]	
	Regression Approach for Composition Ra_4C	4	4.1	[112]	[114, 115]			✓
	s_{sl}Golog	4	4.2		Submitted			
	Web Service Composition Optimization	5	all		[110]			
	Reference Architecture for Composition	7	7.1	In part in [112]	Extension of [30]		Extension of [168], [71, 70]	
Out Of the Ph.D Report	Progression Approach for Composition Pa_4C				[116]			✓
	Semantic Web Service Composition as a Discovery Process				[117]			
	Composition of Telecom Services					[120]		
	Composition of Stateful and Independent Semantic Web Services				[111]			
	Semantic and Syntactic Data Flow in Web Service Composition				[118]			

Table 7.9: Published Works during the Ph.D Thesis.

Automation of Robust Web Service Composition

The main weakness of the robust compositions computation presented in Section 4.1 concern the potential involvement of the end user during the computation of robust Web service compositions hence a supervision of the robust composition process.

Improving Quality of Composition

Coupling Quality of Service and Semantic Links

An interesting future direction consists in studying non functional parameters such as quality of services together with functional parameters to perform composition of Web services. An open issue will be related to the computation and evaluation of the optimal compositions depending on two level parameters.

Coupling Composition and Discovery

Finally it would be interesting to focus on a process that reduces the number of services (ideally to one) in any composition, then reducing the number of semantic links and enhancing the semantic link based composition. This can be performed by discovering more general service that achieve a set of (or cluster of) tasks in the abstract composition. We denote by *Macro composition of Web services* this process.

Experimentation and Evaluation

Last but not of least importance, it could be interesting to apply a efficient discovery component, in order to obtain more relevant and useful semantic links.

Part IV

Appendix

Appendix A

Introduction

Après avoir introduit le contexte dans lequel se situe notre problème de composition de web services, nous présenterons notre modèle de composition lors des chapitres suivants. Chacun des chapitres suivants traitera un élément spécifique du problème de composition de services web[1].

A.1 Contexte

A.1.1 Découverte de web services

Nous proposons d'intégrer en amont de notre processus de composition, le prototype BCov[24] de découverte dynamique de web services sémantiques. La découverte de web services est vue dans BCov comme une nouvelle instance du cadre général de la réécriture de concepts en utilisant une terminologie $\mathcal{T}$, appelée découverte des meilleures couvertures. Étant donnée une requête utilisateur, BCov recherche les sous-ensembles de services qui couvrent au mieux cette requête. La notion de proximité sémantique entre une requête et des services est formellement définie en s'appuyant sur l'opérateur de différence sémantique[198] entre concepts définis en logique de description. Elle consiste à minimiser les différences entre une requête et ses réécritures potentielles, maximisant ainsi l'information commune entre elles. L'intérêt de cette approche réside dans l'utilisation d'un critère de découverte qui est plus souple que les relations de subsomption ou d'équivalence utilisées habituellement dans les approches de découverte de services existantes. De plus, le calcul sous forme conceptuelle des informations communes ainsi que des différences sémantiques entre la requête et les services découverts, rend cette approche très adaptée pour la recherche de services où il n'existe pas d'annuaire centralisé de services.

A.1.2 Composition de web services

Ce travail aura comme point de départ les techniques développées par BCov[24]. Ainsi nous allons étudier la possibilité de composer dynamiquement des web services décrits comme des sources paramétrées de données XML. Afin de répondre à une requête, un ou plusieurs services peuvent être dynamiquement invoqués; c'est l'une des raisons pour laquelle l'exploitation unifiée de données et de services nous parait essentielle. Un point de départ plus orienté sur l'orchestration de services est offert par Bcov. En effet, le fait que Bcov découvre des ensembles de services, et non pas un service isolé à chaque fois, permet de considérer la découverte correspondante comme le point de départ pour de nouvelles compositions. Il faudra par la suite trouver un ordonnancement

[1] Dans la suite document nous confondrons web services, services web et services.

de l'exécution des services découverts tout en tenant compte des contraintes d'orchestration de ces services.

Notre modèle de composition

Lors de cette partie, nous traiterons le problème de *composition de web services* tel un problème de planification de tâches. Résoudre un problème de composition de web services s'identifie à la résolution d'un but précis au moyen de la connaissance d'un état du monde donné (base de connaissance $\mathcal{KB}$). Ainsi un web service composé d'autres services peut être vu comme une boite noire agissant sur l'état du monde avec l'unique but de composer pour mieux résoudre. Notre solution sera influencée par la *"composition au niveau fonctionnel"* introduite lors du chapitre 2. En effet la *"composition au niveau fonctionnel"* a pour but de découvrir les web services correspondants à un certain schéma de planification. Le but étant de rechercher un *"Workflow"* ou *"diagramme d'états"* mettant en jeux les différents services pertinents avec l'objectif de résoudre un but précis.

Cependant, afin de modéliser notre problème de composition tel un problème de planification, il est essentiel de présenter notre critère de sélection de services permettant l'enchaînement de ceux-ci. Ainsi le chapitre E présentera la fonction Sim_T entre deux services comme une combinaison linéaire d'une fonction de comparaison de concepts (chapitre C), et d'une fonction de comparaison de services (chapitre D). Au préalable le chapitre 5B présentera notre modèle général *D4AC* (acronyme de "Decomposition for Automatic Composition"), permettant l'analyse d'une requête utilisateur, et la réduction du problème de composition en sous problèmes plus simples. Le chapitre F aura pour but d'expliciter le processus de construction de la matrice de liens sémantiques, utile pour résoudre le problème de planification. Le chapitre G présentera l'algorithme permettant de construire le plan solution optimal résolvant le problème de composition de web services.

Appendix B

Décomposition du problème de composition

B.1 Introduction

Lors de ce chapitre, nous analyserons l'algorithme permettant la décomposition du problème de composition. Le chapitre est décomposé en trois sections. La deuxième section fait référence aux notations employées. La section B.3 introduit l'algorithme général de composition automatique de web services noté *D4AC*[1]. L'algorithme général sera décomposé en cinq sous parties distinctes. Chacune de ces sous parties sera introduite séparément dans les sous parties de la section B.3.

B.2 Notations

Dans cette section, nous introduisons les notations employées, ainsi que les concepts utiles et nécessaires à la compréhension de l'algorithme général *D4AC*.

B.2.1 Formalisme pour les Web Services

Chaque web service peut contenir les définitions de différentes opérations, identifiables par leur nom, leurs paramètres, et états du monde dans lesquels elles agissent. Pour des raisons de simplicité d'écriture nous confondrons "opération" d'un *Web Service* et *Web Service*. Nous supposons donc que chaque web service représente une unique opération, et chaque opération est représentée par un unique service web. Ainsi une opération d'un *Web Service* ou *Web Service* sera explicité par son nom, la connaissance de ses paramètres (entrées, sorties), ainsi que par ses pré-conditions et effets (i.e: post-conditions). Ainsi nous proposons le formalisme suivant:

$$WebServiceName\ (Parameters,\ WorldState) \tag{B.1}$$

(B.1) permet d'introduire un service web comme une entité ayant des connaissances sur ses paramètres ("Parameters") et sur l'état du monde ("WorldState") dans lequel il peut agir. Les paramètres seront représentés par les entrées et sorties du service web, alors que l'état du monde du service sera représenté par ses pré-conditions et effets. Nous pouvons donc détailler (B.1) au

[1] D4AC: Decomposition for Automatic Composition.

moyen de (B.2):

$$WebServiceName\ (Inputs,\ Outputs,\ PreConditions,\ Effects) \tag{B.2}$$

avec $Inputs$, $Outputs \subset Parameters$ et $PreConditions$, $Effects \subset WorldState$. (B.2) permet d'introduire un service web comme une entité capable de produire un (ou des) résultat(s) concret(s) *Outputs* en fonction d'*Inputs* nécessaire(s). Les *pré-conditions* (i.e PreConditions) renseignent sur l'état dans lequel le monde doit se trouver avant l'invocation d'un service. Les *Effets* (i.e Effect) renseignent sur l'état du monde après l'invocation du service.

Nous pouvons donc définir un service web comme une application $WebServiceName$ définie par:

$$WebServiceName : \begin{cases} Input^n \rightarrow Output^m \\ (i_1, ..., i_n) \rightarrow^{p_s} (o_1, ..., o_m) \end{cases} \tag{B.3}$$

Nous remarquons que $WebServiceName$ possède n paramètres d'entrée notés $i_1, ..., i_n$ et m paramètres de sortie notés $o_1, ..., o_m$. De plus $\rightarrow^{p_s}$ indique la présence de pré-conditions et effets relatifs au service web $WebServiceName$. Ainsi si l'on applique le formalisme (B.3) au service web *FindCinema* (B.4), alors le service web peut être représenté par l'application suivante:

$$FindCinema : \begin{cases} GeographicArea \rightarrow Cinema \\ instance_{Ga} \rightarrow^{p_s} (instance_C) \end{cases} \tag{B.4}$$

où *"FindCinema"* représente un service de recherche de cinémas retournant une instance du concept *Cinema* en fonction d'une instance du concept d'entrée de type *GeographicArea*.

Nous avons délibérément défini (B.1), (B.2) et (B.3) afin d'obtenir un control suffisant sur les entrées, sorties, pré-conditions et effets (IOPE) de chaque service web.

Conditions sur les paramètres (entrées et sorties) du *Web Service*

Les entrées et sorties d'un *Web Service* doivent être identifiables au moyen d'un concept[2] encadré par une ontologie propre. Ainsi les différents paramètres d'une opération d'un *Web service* peuvent être représentés par des instances de différents concepts appartenant à des ontologies différentes. Les paramètres des services web seront représentés au moyen de concepts décrits avec une famille de logique de description de type $\mathcal{ALN}$ ou $\mathcal{FL}_0$. Par exemple le service web *"FindCinema"* fait référence à un unique paramètre d'entrée (noté $instance_{Ga}$) représenté par le concept *"GeographicArea"*, appartenant à l'ontologie *LocationOntology*. Ainsi *"FindCinema"* nécessite une instance du concept "GeographicArea". L'unique paramètre de sortie ($instance_C$) est représenté par une instance du concept *Cinema*, présent dans l'ontologie *CinemaOntology*. *"FindCinema"* retournera une instance du concept "Cinema" si les pré-conditions sont validées. Ainsi les entrées et sorties d'un *Web Service* sont clairement définies en termes de concepts d'une ontologie spécialisée. La conceptualisation des entrées et sorties permettra de faciliter la *"composition au niveau fonctionnel"*.

Conditions sur l'état du monde (pré-conditions et effets) du *Web Service*

Afin de faciliter le raisonnement sur les pré-conditions et effets, ceux-ci seront représentés au moyen de la `Logique des Prédicats du Premier Ordre`. En effet, les pré-conditions et effets des *Web Services* permettent d'estimer l'état du monde lors d'une situation donnée. Il est donc

[2] Élément d'un thesaurus ou d'une ontologie.

essentiel de pouvoir raisonner à l'aide de ces estimateurs du monde. Ainsi nous adoptons le formalisme suivant pour définir l'état du monde (pré-conditions et effets) d'un service web:

$$\begin{cases} PreCondition(WebServiceName, PC_1, ..., PC_n) \leftarrow PC_1Valid(PC_1), ..., PC_nValid(PC_n). \\ Effect(WebServiceName, E_1, ..., E_n). \end{cases} \tag{B.5}$$

(B.5) permet de donner un sens concret à $\rightarrow^{p_s}$. En effet un web service est défini tout d'abord par ses paramètres mais aussi par les états du monde par lesquels il passe. Il est donc nécessaire d'intégrer les pré-conditions et effets dans la définition d'un service web. Par exemple le *Web Service "FindCinema"* déclare deux prédicats *P1* (pré-conditions sur le service) et *E1* (effets sur le service). Ainsi *P1* et *E1* référencés par p_s de $\rightarrow^{p_s}$ sont définis par:

$$\begin{cases} P1(FindCinema,\ instance_{Ga}) \leftarrow Exist(FindCinema), \\ \qquad\qquad\qquad\qquad\qquad\quad GeographicAreaValid(instance_{Ga}). \\ E1(FindCinema,\ instance_c). \end{cases}$$

Ainsi *P1* nous informe que l'opération *FindCinema* doit exister et que le paramètre représenté par une instance de *GeographicArea* doit être valide avant invocation du service. *E1* nous informe que le paramètre de sortie représenté par une instance de *Cinema* est valide après invocation.

B.3 Décomposition pour une meilleure composition

B.3.1 Introduction

Idée générale

Le but de l'algorithme 4 est de proposer un processus de composition de services web. Ainsi l'algorithme a pour objectif de présenter notre approche. L'algorithme compose les services web retrouvés au moyen de *BCov*[24]. La composition est automatisée dans le but de résoudre une requête Q. En effet la composition est guidée par une requête Q, et aidée par une terminologie T et un ensemble de services web $S_{WebServices}$. Notons que chaque requête Q présente trois grandes catégories d'informations:

- ce que veut explicitement l'utilisateur: le but global de la requête;
- ce que veut implicitement l'utilisateur: les buts locaux de la requête;
- ce que connaît l'utilisateur: les instances de certains concepts de la requête;

Le but global de la requête est résolu par composition des buts locaux. L'idée générale est donc de décomposer le but explicite en buts implicites afin de mieux composer. Ainsi chaque sous but (but local) sera résolu indépendamment des autres. Nous traitons donc le problème de composition de services web comme un problème de décomposition. Notre algorithme repose donc sur le paradigme "Diviser pour mieux régner". Une fois la décomposition effectuée, le *"Workflow"* sera créé lors de l'assemblage et de la composition des flows intermédiaires de chaque but résolu. Pour résumer, l'algorithme utilise la *"composition au niveau fonctionnel"* pour résoudre les buts locaux, puis les *flows* intermédiaires proposés seront composés afin d'obtenir un *"Workflow"* cohérent avec la requête Q de l'utilisateur.

Nous proposons dans ce chapitre de tout d'abord découvrir les services web pertinents au moyen de *BCov*[24], puis de décomposer le but global de la requête Q en buts locaux, et enfin

de résoudre chaque but local à l'aide des services découverts. Cette solution consiste à découvrir avant de décomposer. Nous pouvons facilement imaginer une variante de cette solution. En effet, tout d'abord nous décomposerions le but global de la requête Q en buts locaux, puis lors de la création des flows intermédiaires, nous rechercherions les services web pertinents permettant de résoudre le but local. Cette solution consiste à décomposer pour mieux découvrir. La première solution a l'avantage de découvrir les services web une et une seule fois. La deuxième solution a l'avantage d'être plus dynamique mais de complexité plus importante (à cause de l'appel de BCov pour résoudre chaque sous but).

Conditions nécessaires

On suppose que les termes utilisés dans la requête Q sont identifiables au moyen des instances et concepts des ontologies proposées. De plus, les concepts qui représentent les entrées et sorties des services appartiennent à des concepts propres aux ontologies proposées. En outre chaque service web ne référence qu'une et unique opération (nous utiliserons le formalisme (B.1) section B.3 page 211). Enfin, les concepts des ontologies $T_i \subset T$ seront exprimées dans une logique de description de type $\mathcal{ALN}$ ou $\mathcal{FL}_0$[3][91].

B.3.2 Algorithme général

Notre algorithme *D4AC* présente le plan général de la composition de services web. Il se décompose en cinq parties conjointement liées. Chaque partie traite et résoud un nouveau problème lié à la composition automatique de services web. Cet algorithme ne présente que le plan général de la composition de services web retrouvés par *BCov*[24].

Algorithm 4: Décomposition pour une meilleure Composition: *D4AC*

Données: une requête: Q,
un ensemble de services Web: $S_{WebServices} = \{Ws_1, ..., Ws_n\}$,
un ensemble d'ontologies: $T = \{T_1, ..., T_n\}$,
un algorithme de découverte de services web *BCov*.

Résultat: le *Workflow* $\mathscr{W}$ résolvant la requête Q par composition d'éléments de $S_{WebServices}$.

début

(0) *Analyse de la requête Q au moyen de T (traitement de la langue et découverte du but global);*

(1) *Recherche des services web pertinents à la requête Q à l'aide de $BCov(T, Q)$;*

(2) *Analyse du but global de la requête Q et décomposition en buts locaux;*

(3) *Recherche des flows intermédiaires pour chaque but local;*

(4) *Intégration des flows intermédiaires dans le Workflow final $\mathscr{W}$;*

fin

[3]Naturellement la puissance d'expressivité accrue significativement et inévitablement la complexité informatique.

Tout d'abord, nous nous intéresserons aux quatre parties majeures de l'algorithme *D4AC*. Ces quatre parties seront dénotées (0), (2), (3), (4). Comme nous l'avons expliqué en fin de section B.3.1, nous traiterons le problème de découverte de services web en amont de la décomposition en buts locaux. Notons que le processus de décomposition est à la base de notre modèle. En effet, plus nous décomposons le but général, plus nous élaguons l'espace de recherche des services web pertinents pour chaque sous but.

Dans ce chapitre, nous ne détaillerons pas la partie (1) de l'algorithme *D4AC*, qui consiste à découvrir les services web pertinents répondant «au mieux» à la requête Q selon une terminologie T, et une logique de description $\mathcal{ALN}$ ou $\mathcal{FL}_0$ (voir section A.1.1 page 208).

Analyse de la requête Q (0)

Cette partie consiste à rechercher le but global β_G de la requête Q et à construire une base de connaissance $\mathcal{KB}$ afin de découvrir les buts locaux à résoudre. β_G permet de déterminer l'intention de l'utilisateur grâce à la requête Q posée. En effet la connaissance de β_G n'est pas nécessaire à le réussite de *D4AC*. Notons que les buts locaux ne proviennent pas uniquement et nécessairement de la décomposition du but global β_G, mais peuvent aussi être connus à l'aide de $\mathcal{KB}$.

Dans le but de traiter une requête en langue naturelle, nous proposons d'utiliser le modèle proposé par [53, 87] lors du projet *MKBEEM*[4]. En effet [53, 87] permettent la transformation d'une requête utilisateur explicité en langue naturelle en une formule ontologique de type $\mathcal{ALN}$. Cet outil est donc assimilé à un outil de traitement de la langue. Il nous permet donc la construction d'une base de connaissance $\mathcal{KB}$ renseignant sur les concepts et instances présents dans la requête Q.

La connaissance de la base $\mathcal{KB}$ est un atout majeur pour la décomposition en buts locaux.

Ainsi, par exemple la requête Q_{test} suivante renseigne sur la base de connaissance $\mathcal{KB}$ explicitée par le tableau B.1:

Réserver un ticket pour le film recommandé par Tim dans le cinéma le plus proche de ma localisation[5]*, le* 1^{ier} *juillet 2005 à partir de 17 h.*

Concept	Instance
Document	Ticket
Movie	$\emptyset$
Person	Tim
Cinema	$\emptyset$
Location	$\emptyset$
Date	1^{ier} juillet 2005
Hour	17h
PhoneNumber	+33677777777

Table B.1: $\mathcal{KB}$ pour Q_{test}.

[4] Acronyme de Multilingual Knowledge-Based European Electronic Marketplace.
[5] PhoneNumber est une instance connue lors de la requête Q de l'utilisateur.

Analyse du but global de la requête Q et Décomposition en buts locaux (2)

L'algorithme 5 a pour effet la recherche des buts locaux. Cette recherche est aidée par la base de connaissance $\mathcal{KB}$. Cette base permet donc de renseigner sur les instances et concepts connus par le système et couverts par la requête Q. Notons que si des concepts ou instances ne sont pas couverts par $\beta_L V$, alors ceux-ci seront stockés dans l'ensemble des buts locaux non valides noté $\beta_L \backslash \beta_L V$ permettant ainsi la connaissance des buts valides et non valides de $\mathcal{KB}$. La connaissance de $\mathcal{KB}$ a pour effet de décomposer la requête Q en buts locaux valides. Nous supposons dans cette partie qu'il est pertinent de connaître toutes les instances des concepts présents dans la requête Q et donc dans $\mathcal{KB}$. En effet chaque requête posée résoud un problème donné. Mais chaque résolution de problème fait intervenir la résolution de nouveaux sous problèmes. Cette affirmation est légitime puisque le but de la composition est de résoudre un problème général en décomposant ce même problème. Il est donc important et nécessaire de découvrir les sous problèmes engendrés par le problème général. Plus la décomposition en sous problème est fine, plus la composition pourra être efficace. L'algorithme 5 a pour but de découvrir les buts valides $\beta_L V$ qui correspondent à de nouveaux problèmes plus spécifiques. Nous supposons que la découverte de nouveaux problèmes revient à rechercher les instances non initialisées lors de la requête Q. En effet, si une requête générale émet un besoin général, elle émet aussi des besoins plus spécifiques. Ces besoins seront essentiels pour permettre la composition automatique.

Nous proposons d'analyser le tableau B.2 afin de se rendre compte des buts locaux valides découverts à l'aide de l'algorithme 5 et de la requête Q_{test}. L'algorithme 5 propose trois buts locaux à résoudre afin de faciliter le processus de composition automatique. Ces trois sous buts ont pour objectif de résoudre les instances inconnues de concepts connus de la requête Q.

Algorithm 5: Analyse du but global de la requête Q et décomposition en buts locaux (2)

Données: un ensemble d'ontologies: $T = \{T_1, ..., T_n\}$,
une base de connaissance $\mathcal{KB}$.

Résultat: l'ensemble des buts locaux valides $\beta_L V$ et non valides $\beta_L \backslash \beta_L V$

```
début
    βL V ← ∅
    //Décomposition de la requête Q en buts locaux au moyen de KB
    pour chaque instance i de KB faire
        // Validation de l'existence d'une instance du concept représenté par bl ∈ βL
        si i = ∅ alors
            Ajouter ((i, Concept(i)), βL V);
        sinon
            Ajouter ((i, Concept(i)), βL \βL V);
        fin
    fin
fin
```

La décomposition ainsi effectuée permettra l'analyse des buts. On peut supposer que cette connaissance est une aide non négligeable à la composition de services.

Concept	Instance	β_L	$\in \beta_L V$
Document	Ticket		✗
Movie	$\emptyset$	Résoudre Film	✓
Person	Tim		✗
Cinema	$\emptyset$	Résoudre Cinema	✓
Location	$\emptyset$	Résoudre Localisation	✓
Date	1^{ier} juillet 2005		✗
Hour	17h		✗
PhoneNumber	+33677777777		✗

Table B.2: β_L pour Q_{test}.

Recherche des flows intermédiaires pour chaque but local (3)

L'algorithme recherchant les flows intermédiaires permettant de résoudre les buts locaux sera présenté lors des chapitres F et G. Cet algorithme procède par *"chaînage arrière"*. En effet, à partir de buts locaux valides, l'algorithme recherche les flows résolvant les buts intermédiaires. En d'autres termes, à partir des effets escomptés, la partie (3) de l'algorithme *D4AC* recherche les pré-conditions pouvant affecter l'état du monde afin d'atteindre les effets de chaque but local. La recherche des pré-conditions permet de déterminer l'état dans lequel doit être le monde pour pouvoir valider le but. Si toutes les pré-conditions satisfaisant le but local sont vérifiées par le flow intermédiaire alors celui-ci est retourné comme proposition d'un flow intermédiaire. Ce processus est itéré pour chaque but local retrouvé auparavant à l'aide de l'algorithme 5. Notons que la construction des flows intermédiaires sera effectuée par la résolution d'un problème de planification que nous présenterons lors du chapitre G.

Intégration des flows intermédiaires dans le Workflow final (5)

Cet algorithme est basé sur l'heuristique suivante: *afin de finaliser la composition de services web au moyen de la connaissance des flows intermédiaires, nous supposons que le service permettant la composition finale est le service comportant le plus de paramètres d'entrées dans la base de connaissance.* Cependant l'intégration n'est pas l'étape essentielle au processus de composition.

B.4 Synthèse

Nous avons introduit dans ce chapitre l'algorithme général permettant la composition automatique de web services dans le but de résoudre une requête utilisateur Q. Cependant cet algorithme repose sur une construction de flows intermédiaires permettant de résoudre les buts locaux. Afin de détailler cette phase, nous avons délibérément détacher l'algorithme de construction de flows intermédiaires, de l'algorithme général. Ainsi le chapitre C détaillera notre fonction de Matching de concepts. Le chapitre D présentera notre fonction de Matching de services. Le chapitre E construira une fonction de similitude de deux services au moyen de la fonction de Matching de concepts et de la fonction de Matching de services. En effet lors de la construction du plan validant la composition de services, il est fort probable qu'un service donné puisse être composé avec un nombre conséquent de services découverts par BCov. Il est donc important de construire une fonction permettant de donner une valeur de composition entre deux services susceptibles d'être composés. Le chapitre F introduira l'algorithme de construction de la matrice de liens

sémantiques (ou Matrice de Matching). Enfin le chapitre G traitera de l'algorithme permettant la construction de flows intermédiaires à l'aide des buts locaux.

Appendix C

Fonction de comparaison de concepts

C.1 Introduction

Le but d'établir une similarité sémantique de concepts d'une ontologie est de refléter exactement la notion de "proximité" entre ces concepts (i.e. les noeuds sémantiques de l'ontologie représentant ces concepts). Nous parlerons de *"proximité sémantique"* des concepts. Cette métrique nous permettra ainsi de représenter efficacement la notion de proximité sémantique entre deux noeuds (ou concepts) d'une terminologie donnée. Cette métrique aura pour but de permettre la comparaison de concepts de paramètres de différents web services.

C.2 Définition

La notion de distance (métrique) pour les mathématiques est une notion très ancienne provenant de l'époque d'*Aristote* et d'*Euclid*. Une distance δ entre deux points a et b est généralement décrite par les trois axiomes suivants:

1. Définition positive: $\forall a,\ b:\ \delta(a,\ b) \geq 0$ *et* $\delta(a,\ b) = 0$ *si et seulement si* $a = b$;
2. Symétrie: $\delta(a,\ b) = \delta(b,\ a)$;
3. Inégalité triangulaire: $\forall a,\ b,\ c:\ \delta(a,\ b) \leq \delta(a,\ c) + \delta(c,\ b)$.

Nous pouvons par exemple citer une mesure bien connue de distance pour l'espace euclidien, la distance euclidienne, qui est représentée par la somme des différences quadratiques des éléments considérés dans l'espace en question.

C.3 État de l'art

C.3.1 Distance conceptuelle

[172] proposent une métrique appelée *"distance conceptuelle"*. La distance conceptuelle entre deux noeuds (ou concepts) est définie comme le nombre minimal d'arcs séparant deux noeuds (ou concepts). Cette méthode a l'avantage d'expliciter une réelle notion de distance, puisque la distance

conceptuelle vérifie les propriétés de *"Définition positive"*, *"Symétrie"*, et *"Inégalité triangulaire"*. Cette distance possède l'avantage d'expliciter les notions de proximité ou d'éloignement sémantique des concepts. Par exemple (figure C.1), la distance conceptuelle entre "French Cinema" et "Commercial Cinema" est de 1. La distance conceptuelle entre "French Cinema" et "Cinema" est de 2, et la distance conceptuelle entre "French Cinema" et "Familial Cinema" est de 3.

Cette métrique est utilisée par [45] afin de comparer les concepts des paramètres des services web.

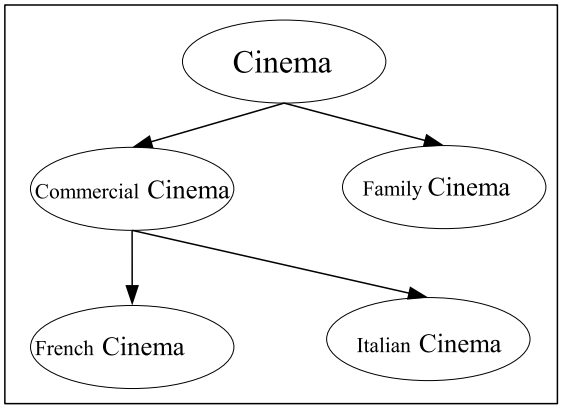


Figure C.1: Taxonomie portant sur le Cinéma.

C.3.2 Probabilité de liens

La méthode de la *"probabilité de liens"* est très proche de la distance conceptuelle de [172]. Cette méthode ne considère pas tous les arcs (ou arrêtes) comme equi-distants. En effet, chaque arc (ou arrête) est pondérée par une *"Probabilité de liens"*. Ainsi la distance en est modifiée. L'intuition de cette métrique est que la distance entre un noeud père et un noeud fils devrait être "plus courte" si la probabilité du noeud père est "proche" de la probabilité du noeud fils.

C.3.3 Couverture de descendance

La métrique de *"couverture de descendance"* d'un lien est définie comme la différence du rapport des descendants subsumés par le noeud père et ceux subsumés par le noeud fils. Cette méthode prend en compte l'ensemble des concepts subsumés par les deux concepts à comparer. Par cette méthode, on explicite la proximité hiérarchique (ou structurelle) des concepts. Ainsi sur la figure C.1, la distance de Couverture de descendance entre "Cinema" et "French Cinema" est:

$$d(Cinema,\ FrenchCinema) = \frac{4}{5} - \frac{0}{5} = \frac{4}{5}$$

La distance de Couverture de descendance entre "Commercial Cinema" et "Cinema" est définie par:

$$d(Cinema,\ CommercialCinema) = \frac{4}{5} - \frac{2}{5} = \frac{2}{5}$$

L'intuition à la base de cette métrique est la suivante: la distance entre un noeud père et fils devrait être "plus faible" si le pourcentage des descendants subsumés par le noeud père est proche

de celui du noeud fils. Cette métrique a été proposée du fait que la majorité des descendants du noeud père sont en général des descendants du noeud fils.

C.3.4 L'opérateur "Différence": un opérateur de soustraction pour la logique de description

[198] décrit l'opérateur de différence à l'aide de la définition 1.

Définition 1. *(opérateur de différence)*
Soit C, D deux descriptions de concepts telles que $C \subseteq D$. La différence $C-D$ noté $diff_{\supseteq}(C,\ D)$ de C et D est définie par $C - D := max_{\supseteq}\{B | B \cap D \equiv C\}$

De plus, il définit la subsomption structurelle comme suit:

Définition 2. *(subsomption structurelle)*
La relation de subsomption d'une logique de description L est dite structurelle si et seulement si pour n'importe quelle clause $A \in L$ et n'importe quelle description $B = B_1 \cap ... \cap B_m \in L$, nous avons l'équivalence suivante $A \supseteq B \Leftrightarrow \exists 1 \leq i \leq m :\ A \supseteq B_i$.

La différence (définie par [198]) entre deux descriptions C et D est définie comme étant une description contenant toute l'information contenue dans une description C mais aucune information de la description de D. Cet opérateur de différence exige que la deuxième opérande subsume la première. Cette métrique est applicable aux concepts décrits dans une logique de description possédant la propriété de subsomption structurelle (par exemple $\mathcal{FL}_0$) afin de raisonner sur les propriétés des concepts.

Cependant cette méthode a le désavantage de ne pas être applicable aux concepts décrits dans une logique de description de forte expressivité (par exemple $\mathcal{ALN}$).

C.3.5 Distance sémantique pour les graphes conceptuels

Étant donnés deux concepts C_1 et C_2 possédant respectivement les types T_1 et T_2, [72] proposent une modification de la distance sémantique de "Sowa" entre C_1 et C_2. L'objectif de cette méthode est de rechercher et trouver le concept C_3 généralisant C_1 et C_2 ayant un type T_3 moins spécifique et moins contraignant qui subsume les types T_1 et T_2. La distance sémantique entre C_1 et C_2 est définie comme la somme des distances de C_1 à C_3 et de C_2 à C_3. Très proche de la distance conceptuelle, cette méthode est plus adaptée aux graphes conceptuels. Il est clair que cette métrique respecte les propriétés de distance, à savoir *"définition positive"*, *"symétrie"* et *"inégalité triangulaire"*.

C.3.6 Matching sémantique

[44] décrivent la similitude entre deux attributs (ou concepts) en utilisant quatre types distincts de sélections, à savoir:

i) le Matching lexicographique;

ii) le Matching sémantique;

iii) le Matching de types;

iv) le Matching structurel.

La similarité lexicographique capture la similitude des noms utilisés pour représenter les concepts (par exemple “FamilyCinema, FCinema”). Ainsi deux concepts auront une forte similitude si ceux-ci contiennent un maximum de sous chaînes lexicales communes. La similarité sémantique se base aussi sur une comparaison lexicale avec un découpage préalable des concepts à comparer afin d'analyser et comparer les sous chaînes de ces deux concepts. Le Matching de types se base sur la relation de subsomption de types, ainsi par exemple (Figure C.1.):

- $M_t(Cinema,\ FamilialCinema) > M_t(FamilialCinema,\ Cinema)$;

Ce type de Matching est proche de la distance sémantique et conceptuelle. Le matching structurel fait référence à une variante de la “Couverture de descendance” (voir section C.3.2 page 219).

Ainsi [72] définissent la notion de similitude entre deux concepts comme un combinaison linéaire des quatre fonctions de Matching prédéfinies auparavant.

C.3.7 Remarques

Il est trivial que la “Distance conceptuelle”, la “Probabilité de liens” et la “Distance sémantique pour les graphes conceptuels” sont assimilables à des distances. En effet ces trois métriques respectent les propriétés de la distance définies auparavant (i.e. *“définition positive”*, *“symétrie”* et *“inégalité triangulaire”*). Cependant les métriques suivantes:

- l'opérateur de soustraction;
- la couverture de descendance;
- le matching sémantique.

ne font pas références à des distances puisque la première est un simple opérateur de comparaison, alors que les deux suivantes ne respectent pas les propriétés de symétrie et d'inégalité triangulaire.

C.4 Notre modèle

C.4.1 Notre métrique de calcul de similitude de concepts

Dans cette sous section, nous décrivons la métrique employée pour comparer deux concepts appartenant à la même ontologie. Notre modèle reposera sur une variante de la “Distance conceptuelle” avec prise en compte de l'opérateur de soustraction $diff_{\supseteq}$. Ainsi nous présentons notre métrique permettant de calculer la similitude de deux concepts appartenant à une même ontologie.

$$Sim_{\supseteq}(C,\ D) = \frac{1}{\alpha+\beta}\left(\frac{\alpha}{1+\#(diff_{\supseteq}(C,\ D))} + \frac{\beta}{1+\delta_c(C,\ D)}\right) \tag{C.1}$$

où $\alpha,\ \beta \in [0,\ 1]$ *et* $\alpha+\beta=1$;
où $diff_{\supseteq}$ représente l' “opérateur Différence”;
où δ_c représente la “Distance conceptuelle”.

Nous n'assimilerons pas $Sim_{\supseteq}(C,\ D)$ à une métrique de distance mais à une métrique de calcul de similitude entre deux concepts. Cette application a pour but d'estimer le degré de similitude entre deux concepts appartenant à une même ontologie, et dont le concept C est subsumé par le concept D. La deuxième condition est imposée par l'opérateur de soustraction.

Cependant, il est possible d'étendre notre métrique à tous concepts C et D (si C et D sont incomparables) d'une même ontologie à condition que la différence descriptive soit définie par:

$$diff_{\supseteq}(C,\ D) := C - lcs(C,\ D)$$

avec *lcs* représentant le *"least common subsumer"*[1]. Nous ne détaillerons pas plus la notion de *"least common subsumer"* puisque nous supposons que les concepts C et D sont comparables (i.e. $C \subseteq D$).

Pour une logique de description de type $\mathcal{ALN}$, la différence descriptive (ou opérateur de difference) peut ne pas prendre en compte les inconsistances non triviales (i.e. celles qui apparaissent dans une conjonction de descriptions non équivalentes à $\bot$), ainsi $Sim_{\supseteq}(C,\ D)$ serait définie avec $(\alpha,\ \beta) = (0,\ 1)$. Cependant notre fonction de similitude peut s'appliquer avec $(\alpha,\ \beta) = (\frac{1}{2},\ \frac{1}{2})$ pour des concepts décrits dans une logique de description de type $\mathcal{FL}_0$.

Paramètres α et β

Les paramètres α et β ont pour rôles de favoriser (ou défavoriser) une des deux métriques introduites dans (C.1) ($diff_{\supseteq}$ ou δ_c). Par exemple, si l'on suppose $(\alpha,\ \beta) = (1,\ 0)$ (resp. $(\alpha,\ \beta) = (0,\ 1)$), alors nous ignorerons la "Distance conceptuelle" (resp. "Différence conceptuelle") au profit de la "Différence conceptuelle" (resp."Distance conceptuelle"). Dans la majorité des cas il est préférable d'utiliser (C.1) avec $(\alpha,\ \beta) = (\frac{1}{2},\ \frac{1}{2})$. α et β représentent des réels de l'intervalle $[0,\ 1]$. Cependant il peut être intéressant de pouvoir personnaliser $Sim_{\supseteq}$ afin de favoriser une métrique donnée.

La Distance conceptuelle $\delta_c(C,\ D)$

Expliquée et détaillée par [100, 172], la distance conceptuelle entre deux noeuds (ou concepts) est définie comme le nombre minimal d'arcs séparant deux noeuds (ou concepts). Ainsi pour deux concepts C et D présents dans une même ontologie $\mathcal{T}$, $\delta_c(C,\ D)$ sera d'autant plus faible (resp. importante) que C et D sont proches (resp. éloignés) dans la taxonomie.

La Différence descriptive $diff_{\supseteq}(C,\ D)$

Expliquée et détaillée par [198], la différence conceptuelle entre deux concepts est définie comme le nombre maximal de propriétés (ou rôles) n'appartenant pas à D mais appartenant à C. Ainsi pour deux concepts C et D présents dans une même ontologie $\mathcal{T}$, $\#(diff_{\supseteq}(C,\ D))$ sera d'autant plus faible (resp. importante) que C et D sont proches (resp. éloignés) en terme de différence conceptuelle.

C.4.2 Exemple

Supposons les trois concepts "Cinema", "FrenchCinema" et "CommercialCinema" appartenant à la même taxonomie $\mathcal{T}$ (Figure E.1 section E.3 page 243). Nous définissons ces trois concepts dans une logique de description $\mathcal{ALN}$ illustrés en figure C.2[2]

Calculons $Sim_{\supseteq}(CommercialCinema,\ Cinema)$ et $Sim_{\supseteq}(FrenchCinema,\ Cinema)$ avec $(\alpha,\ \beta) = (\frac{1}{2},\ \frac{1}{2})$. On obtient respectivement:

[1] Informellement, le *lcs* d'un ensemble de concepts correspond à la description la plus spécifique qui subsume tous les concepts donnés[17].

[2] Les exemples sont délibérement simples pour permettre un bonne compréhension de la fonction de similitude de concepts.

$$
\begin{aligned}
Cinema &\equiv (\geq 1\ hasAddress) \cap (\geq 1\ hasLocation) \cap (\geq 1\ hasName) \\
&\quad \cap (\geq 1\ hasSceancePrice) \\
CommercialCinema &\equiv (\geq 1\ hasAddress) \cap (\geq 1\ hasLocation) \\
&\quad \cap (\geq 1\ hasName) \cap (\geq 1\ hasSceancePrice) \\
&\quad \cap (\geq 15\ hasAdvertisingContract) \\
FrenchCinema &\equiv (\geq 1\ hasAddress) \cap (\geq 1\ hasLocation) \\
&\quad \cap (\geq 1\ hasName) \cap (\forall\ playMovies.French) \\
&\quad \cap (\geq 1\ hasSceancePrice) \\
&\quad \cap (\geq 15\ hasAdvertisingContract) \\
FamilyCinema &\equiv (\geq 1\ hasAddress) \cap (\geq 1\ hasLocation) \\
&\quad \cap (\geq 1\ hasName) \cap (\geq 1\ hasSceancePrice) \\
&\quad \cap (\leq 14\ hasAdvertisingContract) \\
ItalianCinema &\equiv (\geq 1\ hasAddress) \cap (\geq 1\ hasLocation) \\
&\quad \cap (\geq 1\ hasName) \cap (\forall\ playMovies.Italian) \\
&\quad \cap (\geq 1\ hasSceancePrice) \cap (\forall hasEmployee.Italian) \\
&\quad \cap (\geq 15\ hasAdvertisingContract)
\end{aligned}
$$

Figure C.2: Exemple d'une Terminologie $\mathcal{ALN}$ portant sur le Cinéma.

- $Sim_{\supseteq}(CommercialCinema,\ Cinema) = \frac{1}{2} = 0.5$;
- $Sim_{\supseteq}(FrenchCinema,\ Cinema) = \frac{1}{3} \simeq 0.33$.

Ainsi $Sim_{\supseteq}(CommercialCinema,\ Cinema) \geq Sim_{\supseteq}(FrenchCinema,\ Cinema)$, nous pouvons donc conclure que que le concept "CommercialCinema" est plus proche de "Cinema" que "FrenchCinema" au sens de notre métrique.

Si nous calculons le degré de similitude des concepts: *"Cinema"*, *"CommercialCinema"*, *"ItalianCinema"*, *"FrenchCinema"*, et *"FamilyCinema"*, nous obtenons les résultats présentés dans le tableau C.1.

$Sim_{\supseteq}(C,\ D)$	$(\alpha,\ \beta) = (1,\ 0)$			$(\alpha,\ \beta) = (0,\ 1)$			$(\alpha,\ \beta) = (\frac{1}{2},\ \frac{1}{2})$		
	C	CC	IC	C	CC	IC	C	CC	IC
Cinema	1	N/A	N/A	1	N/A	N/A	1	N/A	N/A
CommercialCinema	0.5	1	N/A	0.5	1	N/A	0.5	1	N/A
ItalianCinema	0.25	0.33	1	0.33	0.5	1	0.29	0.41	1
FrenchCinema	0.33	0.5	N/A	0.33	0.5	N/A	0.33	0.5	N/A
FamilyCinema	0.5	N/A	N/A	0.5	N/A	N/A	0.5	N/A	N/A

Table C.1: $Sim_{\supseteq}$ portant sur quelques exemples.

C.5 Synthèse

Nous avons présenté notre fonction de similarité entre deux concepts C et D notée $Sim_{\supseteq}(C,\ D)$. Cette fonction est applicable avec $(\alpha,\ \beta) = (0,\ 1)$ pour des concepts définis dans une logique

de description de type $\mathcal{ALN}$. Pour des concepts définis dans une logique de description de type $\mathcal{FL}_0$, le couple (α, β) n'est soumis à aucune condition.

Appendix D

Fonction de comparaison de services

D.1 Introduction

Dans ce chapitre nous présentons notre fonction de comparaison de services. Ce modèle nécessite la prise en compte de fonctions de Matching (voir tableau D.1) de services afin de composer au mieux les services web. Les fonctions de Matching présentées dans ce tableau sont issues du domaine de l'ingénierie logicielle, qui propose des fonctions de correspondance entre composants logiciels (en termes de leurs entrées et sorties). Ainsi nous comparons le problème de composition de services web avec le problème de composition de composants logiciels. Après avoir introduit le problème de Matching de composants logiciels, nous présenterons une étude comparative des deux problèmes cités ci-dessus. Ensuite nous présenterons une liste de fonctions de Matching permettant la comparaison fonctionnelle de services web. En effet l'enchaînement de deux services web sera vu comme un matching de deux services spécifiques. Ainsi nous introduisons la fonction de comparaison de services.

D.2 Composants et ingénierie logiciels

D.2.1 Introduction

Durant les années 90, la communauté de recherche informatique portant sur l'ingénierie logicielle a consacré de nombreux efforts afin de permettre la réutilisation [185] des composants logiciels. La réutilisation efficace des composants logiciels fiables fournissant une fonctionnalité donnée exige évidemment des moyens efficaces afin de localiser de tels composants. Une approche manuelle, dans laquelle l'ingénieur logiciel doit passer en revue (probablement un grand nombre de) des bibliothèques de composants pour localiser les éléments les plus pertinents est peu appropriée. C'est pourquoi des efforts importants de recherche ont été initiés afin d'obtenir des spécifications formelles de fonctionnalité de composants et de description formelle du composant recherché. Ceci dans le but de permettre la réutilisation automatique des composants pertinents et appropriés. Nous pouvons voir que le problème de recherche automatique de composants logiciels est fortement lié au problème de recherche automatique de services web et donc lié à la composition automatique de services web.

D.2.2 Spécification des fonctions de Matching

La spécification des fonctions de Matching a été proposée dans de nombreux travaux e.g. [104, 185, 207] afin d'évaluer les relations entre composants logiciels et une requête donnée Q représentant les besoins de l'utilisateur. Les fonctions de Matching permettent donc de lier sémantiquement des composants logiciels et une requête donnée Q dans le but d'une découverte automatique et pertinente des composants logiciels. Ainsi s'il y a relation de correspondance entre un composant C et une requête Q, cela signifie que le composant C peut être utilisé pour résoudre le problème schématisé par Q. [88] propose le formalisme suivant:

"la fonctionnalité d'un programme C est spécifiée en termes de pré-conditions initiales C_{pre}[1] (i.e. les conditions qui doivent être vérifiées avant l'exécution du programme C) et de post-conditions C_{post} (i.e. les conditions qui seront vérifiées après l'exécution du programme C)."

La relation entre les conditions préalables et les post-conditions d'un programme donné est donc formulée comme suit:

$$C_{pre}\{Q\}C_{pos}$$

Ce formalisme peut être interprété comme:

"*Si l'affirmation C_{pre} est vraie avant déclenchement d'un programme Q, alors l'affirmation C_{post} sera vraie après son exécution*"[88].

Basée sur ce type d'axiomatisation, la majeure partie des travaux effectués dans l'ingénierie logicielle indique qu'un composant C peut être vu un comme un 2-tuple des attributs (C_{pre}, C_{post}), avec C_{pre} représentant les conditions préalables du composant et C_{post} les post-conditions. De même, une requête Q est spécifiée par un 2-tuple (Q_{pre}, Q_{post}). Les conditions préalables et les post-conditions de la requête donnent une caractérisation du composant désiré en termes de ses conditions préalables et post-conditions.

[207] explorent différentes notions de fonctions de Matching pour rechercher les composants logiciels pertinents et utilisables pour résoudre une requête Q (ou correspondant au mieux aux attentes de Q: en termes des pré-conditions et post-conditions). Toutes les conditions préalables et post-conditions sont représentées par des formules de logique du premier ordre.

D.3 Matching de composants logiciels et Composition de services web

Le problème de Matching de composants logiciels consiste à retrouver un composant logiciel concret C étant donné un composant logiciel abstrait (ou requête) Q dont le degré de correspondance soit maximal. Si l'on suit le formalisme introduit par [104, 185, 207], le problème de Matching de composants logiciels consiste à retrouver un composant C (défini par (C_{pre}, C_{post})) ayant un maximum de correspondances avec le composant Q (défini par (Q_{pre}, Q_{post})). C_{pre} et Q_{pre} définissent respectivement les pré-conditions de C et Q alors que C_{post} et Q_{post} font respectivement référence aux post-conditions de C et Q. Afin de trouver un maximum de correspondances

[1] Les conditions préalables de la requête peuvent être interprétées comme la description des états initiaux du composant recherché.

[104, 185, 207] font référence à treize types de Matching distincts. Les différentes fonctions de Matching font référence à une application $Match$ définie par:

$$Match : \begin{cases} Component * Component \rightarrow \{True, False\} \\ (c_1, c_2) \rightarrow boolean \end{cases}$$

Ainsi l'application notée "$Match$" permet d'évaluer la correspondance des composants logiciels.

Chacune des treize fonctions de Matching présentées par [104, 185, 207] possède sa propre spécificité. Certaines fonctions de Matching favorisent la correspondance des pré-conditions des deux composants à comparer, alors que d'autres favorisent la similarité des post-conditions.

Nous pouvons illustrer une fonction de Matching présentée par [207] notée $M_{exact-pre/post}$ faisant référence à un Matching exact des pré-conditions et post-conditions de la requête "Q" et du composant logiciel "C". Ainsi la fonction $M_{exact-pre/post}$ appliquée à S et Q sera évaluée à $True$ si et seulement si les pré-conditions de la requête Q correspondent complètement avec les pré-conditions du composant retrouvé et si les post-conditions de la requête Q correspondent complètement avec les post-conditions du composant C retrouvé. Toutes les autres fonctions de Matching proposées évaluent différents types de correspondances.

Nous remarquons que le problème de Matching de composants logiciels est relativement proche de notre problème de composition de services web. En effet, composer deux services Ws_1 et Ws_2 revient à retrouver un service web Ws_2 ayant un maximum de similarité[2] avec un service Ws'_1 ayant comme post-conditions les pré-conditions de Ws_1. Cependant afin d'expliquer notre démarche, il est essentiel de faire l'analogie des deux problèmes de façon formelle.

D.3.1 Formalisme pour la composition de services web

Ayant introduit le formalisme d'un web service en page 211, nous pouvons analyser la composition de deux services web Ws_1 et Ws_2[3] comme étant une application $Ws_1 o Ws_2$ définie par:

$$Ws_1 o Ws_2 : \begin{cases} Input_2^n \rightarrow Output_1^l \\ (i_1^2, ..., i_n^2) \rightarrow^{p_s} (o_1^1, ..., o_l^1) \end{cases} \tag{D.1}$$

où Ws_1 et Ws_2 sont respectivement définis par:

$$Ws_1 : \begin{cases} Input_1^m \rightarrow Output_1^l \\ (i_1^1, ..., i_m^1) \rightarrow^{p_s} (o_1^1, ..., o_l^1) \end{cases} \tag{D.2}$$

$$Ws_2 : \begin{cases} Input_2^n \rightarrow Output_2^m \\ (i_1^2, ..., i_n^2) \rightarrow^{p_s} (o_1^2, ..., o_m^2) \end{cases} \tag{D.3}$$

Notons que nous avons uniquement considéré un composition simple de services ne faisant intervenir que deux services. Ce modèle s'étend à des compositions plus complexes (voir Figure D.1). Ainsi pour des raisons de lisibilité et de compréhension nous ne présenterons et n'illustrerons que le problème de composition de deux web services dont les post-conditions de Ws_2 satisfont les pré-conditions de Ws_1. Le cas général s'étend aisément à partir de ce qui a été présenté auparavant.

Nous pouvons donc remarquer que le problème de composition de services web consiste donc à trouver un Matching entre le domaine $Output_2^m$ et le domaine $Input_1^m$. Nous pouvons donc

[2] Voir "Fonction de Similarité" Sim_T page 241.

[3] Nous supposons dans ce cas que toutes les post-conditions de Ws_2 satisfont toutes les pré-conditions de Ws_1.

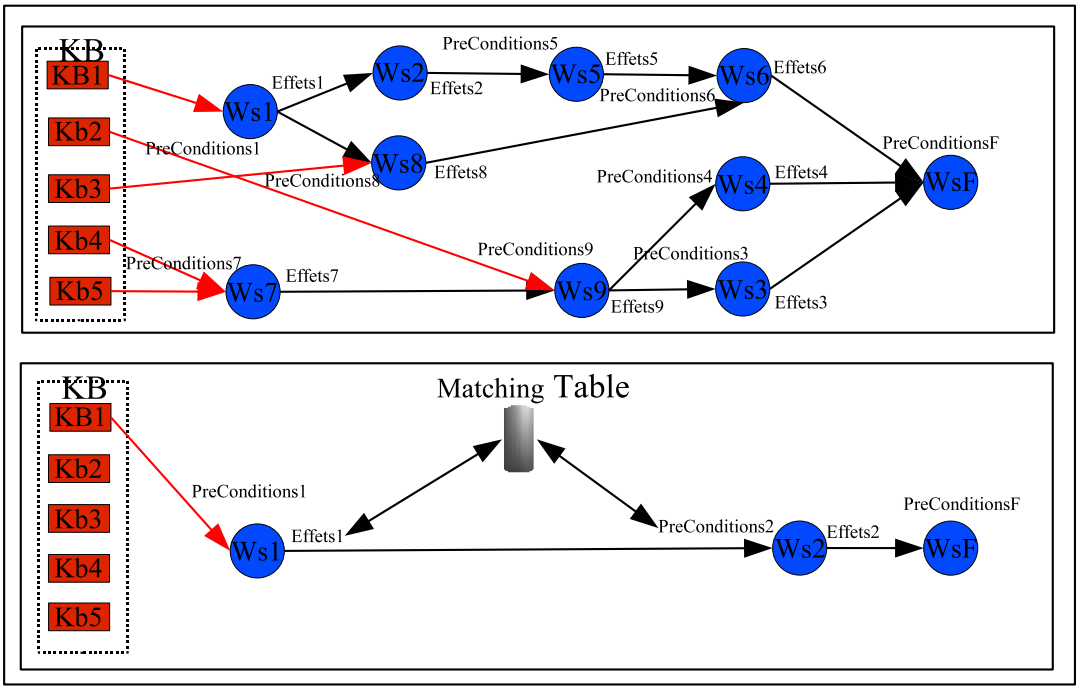


Figure D.1: Composition de deux services et généralisation.

reformuler le problème de composition de services en un problème de Matching de domaines. Ainsi résoudre le problème de composition $Ws_1 o Ws_2$ revient à trouver un Matching entre les m paramètres de sorties (ou post-conditions) $(o_1^2, ..., o_m^2)$ du service web Ws_2 et les m paramètres d'entrées (ou pré-conditions) $(i_1^1, ..., i_m^1)$ du service web Ws_1. Ce problème revient à rechercher un service web Ws_2 ayant des post-conditions correspondant avec les pré-conditions d'un service web Ws_1. Autrement dit résoudre le problème précédant revient à trouver un service web Ws_2 dont les post-conditions correspondent[4] avec les pré-conditions d'un service noté Ws_1' explicité par $Inverse_S(Ws_1)$. Nous définissons l'application $Inverse_S$ par:

$$Inverse_S : \begin{cases} Input^n * Output^m \rightarrow Output^{Card\mathcal{KB}} * Input^n \\ ((i_1, ..., i_n), (o_1, ..., o_m)) \rightarrow^{p_s} ((\mathcal{KB}_1, ..., \mathcal{KB}_{Card\mathcal{KB}}), (i_1, ..., i_n)) \end{cases} \quad \text{(D.4)}$$

où $\mathcal{KB}_i$ représente un élément de la base de connaissance $\mathcal{KB}$. La présence des éléments de $\mathcal{KB}$ dans les pré-conditions de l'image d'un service S par $Inverse_S$ permettra par la suite de favoriser une composition avec des services ayant des pré-conditions déja connues.

Ainsi l'application $Inverse_S$ permet de transformer le problème de composition de services en un problème de Matching de services. Ainsi pour résumer, composer deux services Ws_1 et Ws_2 revient à trouver un Matching adéquat entre $Inverse_S(Ws_1)$ et Ws_2. La figure D.2 explicite le formalisme exploité.

Afin d'expliciter notre nouveau problème, nous introduisons la définition d'un service abstrait de référence pour la composition noté S_{ARC}.

[4]La notion de Correspondance fait référence aux applications $Match$ présentées dans le tableau D.1.

Définition 3. *Un Service Abstrait de Référence pour la Composition noté $S_{ARC}(S)$ est défini comme un service abstrait, image de S par l'application $Inverse_S$. Ainsi les effets de $S_{ARC}(S)$ sont exactement les pré-conditions de S, et les pré-conditions de $S_{ARC}(S)$ sont initialisées aux éléments de $\mathcal{KB}$.*

De même nous définissons l'application $Inverse_S^{i_t}$ restreignant le domaine image du service retourné à i_t.

$$Inverse_S^{i_t} : \begin{cases} Input^n * Output^m \rightarrow Output^{Card\mathcal{KB}} * Input \\ ((i_1, ..., i_n), (o_1, ..., o_m)) \rightarrow^{p_s} ((\mathcal{KB}_1, ..., \mathcal{KB}_{Card\mathcal{KB}}), i_t) \end{cases} \quad \text{(D.5)}$$

Par exemple l'image du service web (B.3) par $Inverse_S^{i_t}$ sera vue comme un service ayant $Card\mathcal{KB}$ pré-conditions $(\mathcal{KB}_1, ..., \mathcal{KB}_C ard\mathcal{KB})$ et une post-condition i_t.

Définition 4. *Un Service Aide à la Composition noté $S_{AIC}^{i_t}(S)$ est défini comme un service abstrait, image de S par l'application $Inverse_S^{i_t}$. Ainsi les effets de $S_{AIC}^{i_t}(S)$ sont réduits au singleton i_t, et les pré-conditions de $S_{AIC}^{i_t}(S)$ sont initialisées aux éléments de $\mathcal{KB}$.*

Remarque: Un web service image de S par l'application S_{ARC} possède plus d'effets qu'un web service image de S par l'application $S_{AIC}^{i_t}$.

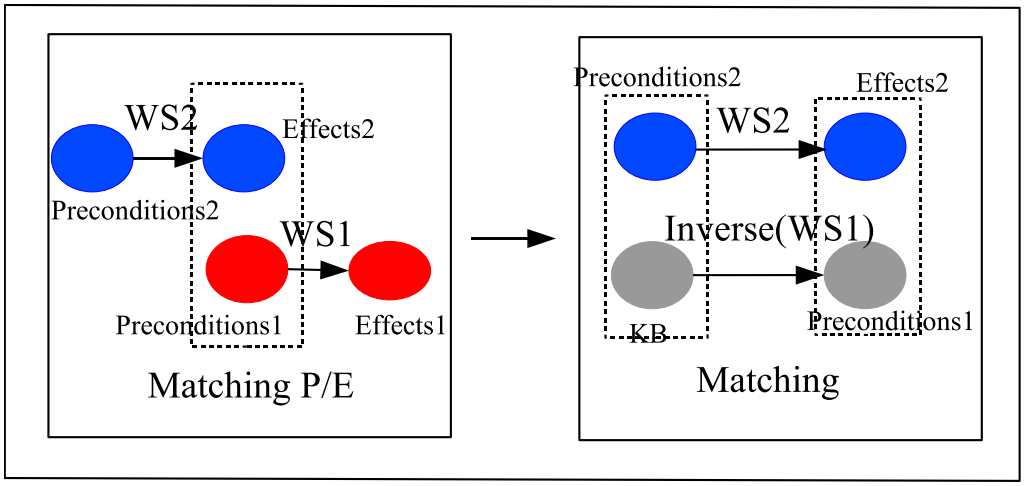


Figure D.2: Transformation du problème de composition en un problème de Matching.

Nous avons donc introduit le problème de Matching de services au niveau fonctionnel (i.e. Pré-conditions/Post-conditions). Comme nous avons introduit un problème de Matching, il est essentiel de proposer des fonctions de Matching afin de pouvoir résoudre le problème de composition de services. Les choix des fonctions de Matching sont présentés dans le tableau D.1. Dans la section suivante nous présenterons les différentes fonctions de Matching utilisées. De plus nous présenterons un ordre partiel sur celles-ci ainsi que deux sous-ordres totaux sur deux sous ensembles distincts.

D.4 Matching et Composition de services web

Dans le but d'obtenir une composition automatisée et pertinente, il est essentiel de composer les services web à l'aide de bonnes propriétés de Matching. [45, 106, 152] proposent de composer deux services Ws_1 et Ws_2 si et seulement si les pré-conditions de Ws_1 (resp Ws_2) correspondent [5] avec les post-conditions de Ws_2. Ceci dans un but d'obtenir un composition du type $Ws_1 o Ws_2$. Cependant leur modèle ne repose que sur trois types de Matching (basé sur la subsomption de concepts). Adopter un tel modèle ne permet malheureusement pas d'appliquer un grand nombre de fonctions de Matching. C'est pourquoi nous redéfinissons le problème de composition de services afin de concevoir plus de trois types de fonctions de Matching. En effet la nouvelle définition du problème permet de prendre en compte plus de paramètres que dans les cas proposés par [45, 106, 152] (notamment la prise en compte des pré-conditions de Ws_1 et des post-conditions de Ws_2 pour une composition du type $Ws_1 o Ws_2$). Leur modèle ne repose que sur la relation de subsomption des post-conditions de Ws_2 et des pré-conditions de Ws_1. Cependant en prenant en compte plus de paramètres, nous nous exposons à une multiplicité des fonctions de Matching. Il est donc important de personnaliser nos fonctions de Matching pour obtenir une table de Matching pertinente, non redondante et utile à une composition efficace de services web.

Dans cette section, nous analyserons chacune des fonctions de Matching proposées et introduirons les spécificités de chacune. Basées sur la logique propositionnelle, nous présentons les fonctions de Matching étudiées dans la table D.1 (Domaine des services web).

Match(S, Q)	Définition
1.$M_{exact-pre/post}$(S,Q)	$(S_{pre} \leftrightarrow Q_{pre}) \wedge (Q_{post} \leftrightarrow S_{post})$
2.$M_{exact-pre-\alpha}$(S,Q)	$(S_{pre} \leftrightarrow Q_{pre}) \wedge (Q_{post} \rightarrow S_{post})$
3.$M_{exact-post-\alpha}$(S,Q)	$(S_{pre} \rightarrow Q_{pre}) \wedge (Q_{post} \leftrightarrow S_{post})$
4.$M_{plug-in}(S, Q)$	$(S_{pre} \rightarrow Q_{pre}) \wedge (Q_{post} \rightarrow S_{post})$
5.$M_{plug-in-post}(S, Q)$	$(Q_{post} \rightarrow S_{post})$

Table D.1: Résumé des fonctions de matching proposées.

Tout d'abord, remarquons que les fonctions de Matching 1, 4 et 5 ont été étudiées dans le domaine des composants logiciels (i.e. $C \leftrightarrow S$ où S dénote un composant logiciel). Nous ne détaillerons pas les modifications employées. En effet quelques modifications ont dû être effectuées afin que ces fonctions de Matching aient un sens dans le domaine des services web. À ces trois premières fonctions de Matching, nous avons ajouté deux fonctions de Matching distinctes et pertinentes pour notre domaine d'application notées:

- $M_{exact-pre-\alpha}$;
- $M_{exact-post-\alpha}$.

Ces deux différentes fonctions de Matching permettent d'obtenir un control plus fin sur les pré-conditions et effets des services web à retrouver. Nous pouvons noter que notre problème de composition de services web s'est partiellement transformé en un problème de découverte de services web (i.e. via un problème de Matching de services web). En effet, dans un but de composer les services les plus pertinents, nous procédons par découverte de services au moyen de fonctions de Matching.

[5] Pour définir Correspondance, [45, 106, 152] définissent trois types de Matching: ExactMatch, GenericMatch et SubsumeMatch. [45] définit un quatrième type de Matching basé sur la comparaison de concepts n'appartenant pas à une même ontologie.

Pour l'illustration des différentes fonctions de Matching proposées, nous raisonnerons sur les concepts et non sur les pré-conditions et post-conditions. Ceci pour des raisons de compréhension et de lisibilité. Les différentes fonctions de matching proposées raisonnent sur les pré-conditions et post-conditions des services web. Celles-ci sont donc exprimées avec la logique des prédicats du premier ordre.

D.4.1 Détails des différentes fonctions de Matching

La fonction de Matching $M_{exact-pre/post}$

[207] identifie la fonction de Matching exacte (1^{ere} ligne du tableau D.1) entre un service S et un service Q comme l'équivalence des pré-conditions de Q et S ainsi que l'équivalence des post-conditions Q et S. Ainsi les effets désirés de Q sont vérifiés par le service S puisque équivalents. Puisque l'équivalence entre les composantes fonctionnelles de Q et S est une condition forte, il est nécessaire de prendre en compte des notions de fonctions de Matching beaucoup plus souples. Cependant, il n'est pas nécessaire d'avoir une équivalence totale entre les pré-conditions de Q et S puisque le problème de composition est en fait essentiellement renfermé dans les effets de Q et S.(voir section D.3.1 page 227).

Dans les exemples qui suivent, nous supposons les relations de subsomption de la figure D.3 (ou figure E.1).

$DistrictCinema \subseteq Cinema \subseteq EntertainementPlace$
$Location \subseteq GeographicArea$

Figure D.3: Quelques relations de subsomption.

1. Exemple:

   Si l'on suppose trois services distincts tels que:

   - Ws_1(pre Location, post Cinema);
   - Ws_2(pre Location, post Cinema);
   - Ws_3(pre GeographicArea, post Cinema).

   Nous pouvons remarquer que la fonction de Matching $M_{exact-pre/post}$ appliquée à Ws_1 et Ws_2 retournera *"True"*, alors que la même fonction retournera *"False"* pour Ws_2 et Ws_3 (resp Ws_1 et Ws_3).

2. Exemple:

   Si l'on suppose trois services distincts tels que:

   - Ws_1(pre Cinema, post Price);
   - Ws_2(pre Location, post Cinema);
   - Ws_3(pre Location, post GeographicArea).

   avec $\mathcal{KB} = Location$.

   Nous pouvons remarquer que la fonction de Matching $M_{exact-pre/post}$ appliquée à $Inverse_S^{Cinema}(Ws_1)$ et Ws_2 retournera *"True"*, alors que le même Matching retournera

"False" pour Ws_2 et Ws_3 (resp Ws_1 et Ws_3). Ainsi nous pouvons conclure que Ws_1 et Ws_2 sont composables via la composition $Ws_1 o Ws_2$ puisque $M_{exact-pre/post}(Inverse_S^{Cinema}(Ws_1),\ Ws_2) = True$.

La fonction de Matching $M_{exact-pre-\alpha}$

Dans $M_{exact-pre-\alpha}$ (2^{ieme} ligne du tableau D.1), Q sera assimilé à des services web ayant des pré-conditions équivalentes et des post-conditions plus fortes i.e. les post-conditions du service recherché Q impliquent les post-conditions du service S.

1. Exemple:

   Si l'on suppose deux services distincts tels que:

   - Ws_3(pre GeographicArea, post Cinema);
   - Ws_4(pre GeographicArea, post EntertainementPlace).

   Nous pouvons remarquer que la fonction de Matching $M_{exact-pre-\alpha}$ appliquée à Ws_4 et Ws_3 retournera *"True"* si $Q = Ws_3$ et $S = Ws_4$, alors que le même Matching retournera *"False"* pour $Q = Ws_4$ et $S = Ws_3$.

2. Exemple:

   Si l'on suppose deux services distincts tels que:

   - Ws_3(pre EntertainementPlace, post Price);
   - Ws_4(pre Location, post Cinema).

   avec $\mathcal{KB} = Location$.

   Nous pouvons remarquer que la fonction de Matching $M_{exact-pre-\alpha}$ appliquée à $Inverse_S^{EntertainementPlace}(Ws_3)$ et Ws_4 retournera *"True"*, alors que le même Matching retournera *"False"* pour $Inverse_S^{Location}(Ws_4)$ et Ws_3. Ainsi nous pouvons conclure que Ws_3 et Ws_4 sont composables via la composition $Ws_3 o Ws_4$ puisque $M_{exact-post-\alpha}(Inverse_S^{EntertainementPlace}(Ws_3),\ Ws_4) = True$.

La fonction de Matching $M_{exact-post-\alpha}$

Dans $M_{exact-post-\beta}$ (3^{ieme} ligne du tableau D.1), Q sera assimilé avec des services web ayant des pré-conditions plus fortes et des post-conditions équivalentes i.e. les conditions préalables du service S impliquent les conditions préalables du service recherché Q.

1. Exemple:

   Si l'on suppose deux services distincts tels que:

   - Ws_1(pre Location, post Cinema);
   - Ws_5(pre GeographicArea, post Cinema).

   Nous pouvons remarquer que la fonction de Matching $M_{exact-post-\alpha}$ appliquée à Ws_1 et Ws_5 retournera *"True"* si $Q = Ws_5$ et $S = Ws_1$, alors que la même fonction retournera *"False"* pour $Q = Ws_1$ et $S = Ws_5$.

2. Exemple:

   Si l'on suppose deux services distincts tels que:

   - Ws_1(pre Cinema, post Price);
   - Ws_5(pre GeographicArea, post Cinema).

   avec $\mathcal{KB} = Location$.

   Nous pouvons remarquer que la fonction de Matching $M_{exact-post-\alpha}$ appliquée à $Inverse_S^{Cinema}(Ws_1)$ et Ws_5 retournera *"True"*, alors que la même fonction retournera *"False"* pour Ws_5 et $Inverse_S^{Cinema}(Ws_1)$. Ainsi nous pouvons conclure que Ws_1 et Ws_5 sont composables via la composition $Ws_1 o Ws_5$ puisque $M_{exact-post-\alpha}(Inverse_S^{Cinema}(Ws_1),\ Ws_5) = True$.

La fonction de Matching $M_{plug-in}$

Pour la fonction de Matching Plug-in Match (4^{ieme} ligne du tableau D.1), Q sera assimilé avec des services web ayant des pré-conditions plus fortes et des post-conditions plus faibles i.e. les post-conditions du service recherché Q impliquent (et satisfont, en conséquence) les post-conditions du service S, et les conditions préalables du service S impliquent les conditions préalables du service recherché Q.

1. Exemple:

   Si l'on suppose trois services distincts tels que:

   - Ws_1(pre GeographicArea, post Cinema);
   - Ws_3(pre GeographicArea, post Cinema);
   - Ws_4(pre Location, post EntertainementPlace).

   Nous pouvons remarquer que la fonction de Matching Plug-in (ou "generic Match") $M_{plug-in}$ appliquée à Ws_1 et Ws_4 retournera *"True"* si $Q = Ws_1$ et $S = Ws_4$, alors que la même fonction retournera *"False"* pour $Q = Ws_4$ et $S = Ws_1$.

2. Exemple:

   Si l'on suppose trois services distincts tels que:

   - Ws_1(pre EntertainementPlace, post Price);
   - Ws_3(pre GeographicArea, post Cinema);
   - Ws_4(pre Location, post Cinema).

   avec $\mathcal{KB} = Location$.

   Nous pouvons remarquer que la fonction de Matching $M_{plug-in}$ appliquée à $Inverse_S^{EntertainementPlace}(Ws_1)$ et Ws_4 retournera *"True"*, alors que la même fonction retournera *"False"* pour Ws_4 et Ws_3 (resp Ws_1 et Ws_3). Ainsi nous pouvons conclure que Ws_1 et Ws_4 sont composables via la composition $Ws_1 o Ws_4$ puisque $M_{plug-in}(Inverse_S^{EntertainementPlace}(Ws_1),\ Ws_4) = True$.

La fonction de Matching $M_{plug-in-post}$

La fonction de Matching $M_{plug-in-post}$ (5^{ieme} ligne du tableau D.1) ne considère pas les conditions préalables pour déterminer un Matching. Dans ce Matching nous nous intéressons seulement aux résultats de l'exécution du service. Dans ce cas, la relation qui doit être validée entre S et Q concerne la relation sur les post-conditions de S et Q. Les post-conditions de Q doivent impliquer les post-conditions de S i.e. la garantie des post-conditions de S. Cette notion est également considérée dans [164].

1. Exemple:

   Si l'on suppose trois services distincts tels que:

   - Ws_1(pre Location, post Cinema)
   - Ws_4(pre GeographicArea, post EntertainementPlace);
   - Ws_3(pre GeographicArea, post Cinema).

   Nous pouvons remarquer que la fonction de Matching $M_{plug-in-post}$ appliquée à Ws_1 et Ws_4 retournera *"True"* si $Q = Ws_1$ et $S = Ws_4$, alors que la même fonction retournera *"False"* pour $Q = Ws_4$ et $S = Ws_1$.

2. Exemple:

   Si l'on suppose trois services distincts tels que:

   - Ws_1(pre EntertainementPlace, post Price)
   - Ws_4(pre GeographicArea, post EntertainementPlace);
   - Ws_3(pre GeographicArea, post Cinema).

   avec $\mathcal{KB} = Location$.

   Nous pouvons remarquer que la fonction de Matching $M_{plug-in-post}$ appliquée à $Inverse_S^{EntertainementPlace}(Ws_1)$ et Ws_4 retournera *"True"*, alors que la même fonction retournera *"False"* pour $Inverse_S^{GeographicArea}(Ws_4)$ et Ws_3. Ainsi nous pouvons conclure que Ws_1 et Ws_4 sont composables via la composition $Ws_1 o Ws_4$ puisque $M_{exact-post-\alpha}(Inverse_S^{EntertainementPlace}(Ws_1),\ Ws_4) = True$.

D.4.2 Construction d'un ordre partiel sur les fonctions de Matching

Après avoir introduit un certain nombre de fonctions de Matching utiles et nécessaires pour la composition de services web, il est essentiel de pouvoir ordonner ces fonctions de Matching afin de déterminer quelles fonctions sont plus générales ou spécifiques. Ainsi, dans un but de construire un ordre partiel sur les fonctions de Matching proposées dans le tableau D.1, nous introduisons le théorème suivant.

Theorème 1. *Étant données les cinq relations de Matching introduites dans la tableau D.1, nous avons les relations suivantes:*

i) $M_{exact-pre/post}\ \mathcal{R}\ M_{exact-pre-\alpha}\ \mathcal{R}\ M_{plug-in}\ \mathcal{R}\ M_{plug-in-post}$

ii) $M_{exact-pre/post}\ \mathcal{R}\ M_{exact-post-\alpha}\ \mathcal{R} M_{plug-in}$

avec $\mathcal{R}$ représentant une relation binaire (i.e. l'implication logique notée $\rightarrow$).

Proof. La preuve du Théorème 1 se décompose en cinq étapes. Ainsi nous allons décomposer i) et ii) et prouver chacune des cinq implications.

1. Prouvons tout d'abord que $M_{exact-pre/post} \rightarrow M_{exact-pre-\alpha}$ est vraie,
   i.e $(S_{pre} \leftrightarrow Q_{pre}) \wedge (Q_{post} \leftrightarrow S_{post}) \rightarrow (S_{pre} \leftrightarrow Q_{pre}) \wedge (Q_{post} \rightarrow S_{post})$ est vraie
   i.e $(Q_{post} \leftrightarrow S_{post}) \rightarrow (Q_{post} \rightarrow S_{post})$ est vraie
   i.e $\neg(Q_{post} \leftrightarrow S_{post}) \vee (Q_{post} \rightarrow S_{post})$ est une tautologie

   Ainsi montrons que $(\neg(Q_{post} \leftrightarrow S_{post}) \vee (Q_{post} \rightarrow S_{post}))$ est une tautologie
   $(\neg(Q_{post} \leftrightarrow S_{post}) \vee (Q_{post} \rightarrow S_{post}))$
   $\Leftrightarrow (\neg((\neg Q_{post} \vee S_{post}) \wedge (\neg S_{post} \vee Q_{post})) \vee (\neg Q_{post} \vee S_{post}))$
   $\Leftrightarrow (\neg((\neg Q_{post} \wedge \neg S_{post}) \vee (\neg Q_{post} \wedge Q_{post}) \vee (S_{post} \wedge \neg S_{post}) \vee (S_{post} \wedge Q_{post})) \vee (\neg Q_{post} \vee S_{post})$
   $\Leftrightarrow (\neg((\neg Q_{post} \wedge \neg S_{post}) \vee (Q_{post} \wedge S_{post}))) \vee (\neg Q_{post} \vee S_{post})$
   $\Leftrightarrow ((Q_{post} \vee S_{post}) \wedge (\neg Q_{post} \vee \neg S_{post})) \vee (\neg Q_{post} \vee S_{post})$
   $\Leftrightarrow ((Q_{post} \wedge \neg S_{post}) \vee (Q_{post} \wedge \neg Q_{post}) \vee (S_{post} \wedge \neg S_{post}) \vee (S_{post} \wedge \neg Q_{post})) \vee (\neg Q_{post} \vee S_{post})$
   $\Leftrightarrow ((Q_{post} \wedge \neg S_{post}) \vee (\neg Q_{post} \wedge S_{post})) \vee (\neg Q_{post} \vee S_{post})$
   $\Leftrightarrow ((Q_{post} \wedge \neg S_{post}) \vee (\neg Q_{post} \wedge S_{post})) \vee (\neg Q_{post} \wedge (S_{post} \vee \neg S_{post})) \vee (S_{post} \wedge (\neg Q_{post} \vee Q_{post}))$
   $\Leftrightarrow ((Q_{post} \wedge \neg S_{post}) \vee (\neg Q_{post} \wedge S_{post})) \vee (\neg Q_{post} \wedge \neg S_{post}) \vee (Q_{post} \wedge S_{post})$
   $\Leftrightarrow (Q_{post} \wedge (S_{post} \vee \neg S_{post})) \vee (\neg Q_{post} \wedge (S_{post} \vee \neg S_{post})))$
   $\Leftrightarrow Q_{post} \vee \neg Q_{post}$
   $\Leftrightarrow True$

2. Prouvons que $M_{exact-pre-\alpha} \rightarrow M_{plug-in}$ est vraie,
   i.e $((S_{pre} \leftrightarrow Q_{pre}) \wedge (Q_{post} \rightarrow S_{post})) \rightarrow (S_{pre} \rightarrow Q_{pre}) \wedge (Q_{post} \rightarrow S_{post})$ est vraie
   i.e $(S_{pre} \leftrightarrow Q_{pre}) \rightarrow (S_{pre} \rightarrow Q_{pre})$ est vraie
   i.e $\neg(S_{pre} \leftrightarrow Q_{pre}) \vee (S_{pre} \rightarrow Q_{pre})$ est une tautologie
   Or cette proposition est bien une tautologie, puisque démontrée dans 1).

3. Prouvons que $M_{plug-in} \rightarrow M_{plug-in-post}$ est vraie,
   i.e $((S_{pre} \rightarrow Q_{pre}) \wedge (Q_{post} \rightarrow S_{post})) \rightarrow (Q_{post} \rightarrow S_{post})$ est vraie
   i.e $\neg((S_{pre} \rightarrow Q_{pre}) \vee (Q_{post} \rightarrow S_{post})) \vee (Q_{post} \rightarrow S_{post})$ est une tautologie

   Ainsi montrons que $\neg((S_{pre} \rightarrow Q_{pre}) \vee (Q_{post} \rightarrow S_{post})) \vee (Q_{post} \rightarrow S_{post})$ est une tautologie
   $\neg((S_{pre} \rightarrow Q_{pre}) \vee (Q_{post} \rightarrow S_{post})) \vee (Q_{post} \rightarrow S_{post})$
   $\Leftrightarrow \neg((\neg S_{pre} \vee Q_{pre}) \vee (\neg Q_{post} \vee S_{post})) \vee (Q_{post} \rightarrow S_{post})$
   $\Leftrightarrow (S_{pre} \wedge Q_{pre}) \vee (Q_{post} \vee \neg S_{post}) \vee (\neg Q_{post} \vee S_{post})$
   $\Leftrightarrow (S_{pre} \wedge Q_{pre}) \vee (Q_{post} \vee \neg S_{post}) \vee (\neg Q_{post} \wedge (S_{post} \vee \neg S_{post})) \vee (S_{post} \wedge (Q_{post} \vee \neg Q_{post}))$
   $\Leftrightarrow (S_{pre} \wedge Q_{pre}) \vee (Q_{post} \vee \neg S_{post}) \vee (\neg Q_{post} \vee S_{post}) \vee (\neg Q_{post} \vee \neg S_{post}) \vee (Q_{post} \vee S_{post})$
   $\Leftrightarrow (S_{pre} \wedge Q_{pre}) \vee (Q_{post} \wedge (S_{post} \vee \neg S_{post})) \vee (\neg Q_{post} \wedge S_{post} \vee \neg S_{post})$
   $\Leftrightarrow (S_{pre} \wedge Q_{pre}) \vee (Q_{post} \vee \neg Q_{post})$
   $\Leftrightarrow True$

4. Prouvons que $M_{exact-pre/post} \rightarrow M_{exact-post-\alpha}$ est vraie,
   i.e $((S_{pre} \leftrightarrow Q_{pre}) \wedge (Q_{post} \leftrightarrow S_{post})) \rightarrow ((S_{pre} \rightarrow Q_{pre}) \wedge (Q_{post} \leftrightarrow S_{post}))$ est vraie
   i.e $(S_{pre} \leftrightarrow Q_{pre}) \rightarrow (S_{pre} \rightarrow Q_{pre})$ est vraie
   i.e $\neg(S_{pre} \leftrightarrow Q_{pre}) \vee (S_{pre} \rightarrow Q_{pre})$ est une tautologie
   Or cette proposition est bien une tautologie, puisque démontrée dans 2).

5. Prouvons que $M_{exact-post-\alpha} \rightarrow M_{plug-in}$est vraie,
   i.e $((S_{pre} \rightarrow Q_{pre}) \wedge (Q_{post} \leftrightarrow S_{post})) \rightarrow (S_{pre} \rightarrow Q_{pre}) \wedge (Q_{post} \rightarrow S_{post}))$ est vraie,

i.e $(Q_{post} \leftrightarrow S_{post}) \rightarrow (Q_{post} \rightarrow S_{post})$ est vraie
i.e $\neg(Q_{post} \leftrightarrow S_{post}) \vee (Q_{post} \rightarrow S_{post})$ est une tautologie
Or cette proposition est bien une tautologie, puisque démontrée dans 1).

□

Le théorème 1 nous permet de construire un ordre partiel sur les fonctions de Matching proposées. Cet ordre repose sur la relation d'implication notée $\rightarrow$ (voir figure D.3). Cependant, dans un but de comparer toutes les fonctions de Matching proposées, il est nécessaire d'avoir un ordre total. Ainsi nous décomposons l'ordre partiel introduit en deux sous ordres totaux. Ainsi nous définissons deux sous ordres totaux sur deux ensembles distincts de fonctions de Matching notés E et F. Ainsi nous notons E (et respectivement F) comme suit:

$$E = \{M_{exact-pre/post},\ M_{exact-pre-\alpha}\ M_{plug-in}\ M_{plug-in-post}\}$$

$$F = \{M_{exact-pre/post},\ M_{exact-post-\alpha},\ M_{plug-in},\ M_{plug-in-post}\}$$

E et F représentent deux ensembles finis de fonctions de Matching.

Une fois introduit E et F, nous définissons l'ordonnancement des éléments de E et F comme suit:

- $M_{exact-pre/post} \rightarrow M_{exact-pre-\alpha} \rightarrow M_{plug-in} \rightarrow M_{plug-in-post}$
- $M_{exact-pre/post} \rightarrow M_{exact-post-\alpha} \rightarrow M_{plug-in} \rightarrow M_{plug-in-post}$

Propriété 1. *La relation d'ordre $\rightarrow$ sur E et F est une relation d'ordre total.*

Proof. La relation d'ordre $\rightarrow$ sur E et F est une relation d'ordre total si deux éléments quelconques x et y de E (resp F) sont comparables, c'est à dire si nous avons une des deux situations suivantes: $x \rightarrow y$ ou bien $y \rightarrow x$. La relation $\rightarrow$ étant transitive, nous avons une relation $\rightarrow$ entre chaque élément de E ou F. □

Propriété 2. *La relation d'ordre $\rightarrow$ sur $E \cup F$ est une relation d'ordre partiel.*

Proof. Procédons par contraposée. Supposons que la relation d'ordre $\rightarrow$ sur $E \cup F$ est une relation d'ordre total. Alors $\forall(x,\ y) \in E \cup F$, nous avons une des deux possibilités suivantes:

- $x \rightarrow y$;
- ou $y \rightarrow x$.

Cependant $\exists(a,\ b) \in E \cup F$ *tel que* $a \nrightarrow b$ *et* $b \nrightarrow a$. En effet $a = M_{exact-pre-\alpha}$ et $b = M_{exact-post-\alpha}$ ne sont pas comparables. Nous avons donc montré qu'il existe au moins un couple $(a,\ b) \in E \cup F$ où a et b ne sont pas comparables. La relation d'ordre $\rightarrow$ sur $E \cup F$ est donc une relation d'ordre partiel. □

La figure D.3 explicite les deux ordres totaux (et donc l'ordre partiel) sur E et F. Nous avons décomposé $E \cup F$ afin de pouvoir comparer et ordonnancer les fonctions de Matching proposées. En effet, nous ne pouvons pas comparer deux services web à l'aide d'une seule fonction de Matching. Il est donc nécessaire de proposer plus d'une fonction de Matching. L'augmentation de ce nombre de fonctions de Matching réduit considérablement le non déterminisme au niveau de la composition des services web. Plus nous avons de fonctions de Matching et plus les services web auront des possibilités de correspondre entre eux, en fonction des spécificités des fonctions de matching. Nous avons défini cinq fonctions différentes de Matching mais il serait envisageable d'étendre ce nombre afin de pouvoir faire correspondre un maximum de services et de pouvoir pondérer chaque couple de services web.

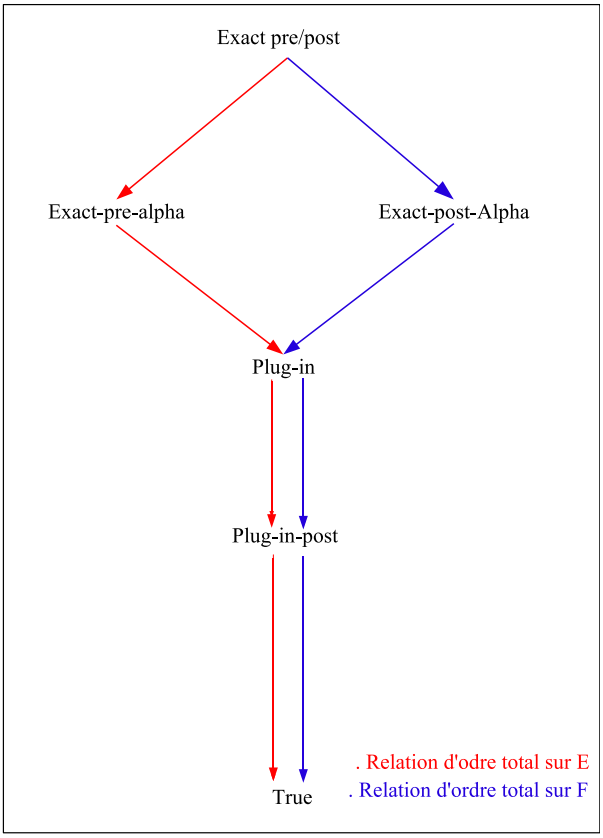


Figure D.4: Treillis des fonctions de Matching.

D.4.3 Pondération des fonctions de Matching

D.4.4 Introduction de f_p

Après avoir introduit les différents types de fonctions de Matching nécessaires pour la composition de services, nous construisons une fonction de pondération notée f_p permettant de pondérer les différentes fonctions de Matching. La figure D.4 explicite la fonction de pondération f_p des fonctions de Matching. La pondération des fonctions de Matching est faite en accord avec l'ordre partiel sur les fonctions de Matching. Ainsi nous définissons f_p comme une application:

$$f_p : \begin{cases} MatchingFunction \rightarrow [0,\ 1] \\ m_f \rightarrow 0.8\ if\ m_f = M_{exact-pre/post} \\ \qquad 0.6\ if\ m_f = M_{exact-pre-\alpha}\ or\ m_f = M_{exact-post-\alpha} \\ \qquad 0.4\ if\ m_f = M_{plug-in} \\ \qquad 0.2\ if\ m_f = M_{plug-in-post} \\ \qquad 0\ otherwise \end{cases} \tag{D.6}$$

Analysons l'application f_p décrite en (D.6) et Figure D.4. f_p est une application définie de $MatchingFunction$ dans [0, 1], ainsi à chaque fonction de Matching décrite précédemment, est

associée une valeur réelle de l'intervalle [0, 1]. Notons que plus la fonction de Matching M est contraignante, plus l'image de M par f_p sera élevée. Ainsi deux services Ws_1 et Ws_2 ayant une valeur de vérité "Vraie" pour la fonction de Matching $M_{exact-pre/post}$ auront un poids de correspondance $f_p(M_{exact-pre/post}) = 0.8$ correspondant à la valeur maximale. Cette application f_p est essentielle pour permettre la comparaison des fonctions de Matching. Notons que f_p est assimilée à une application linéaire.

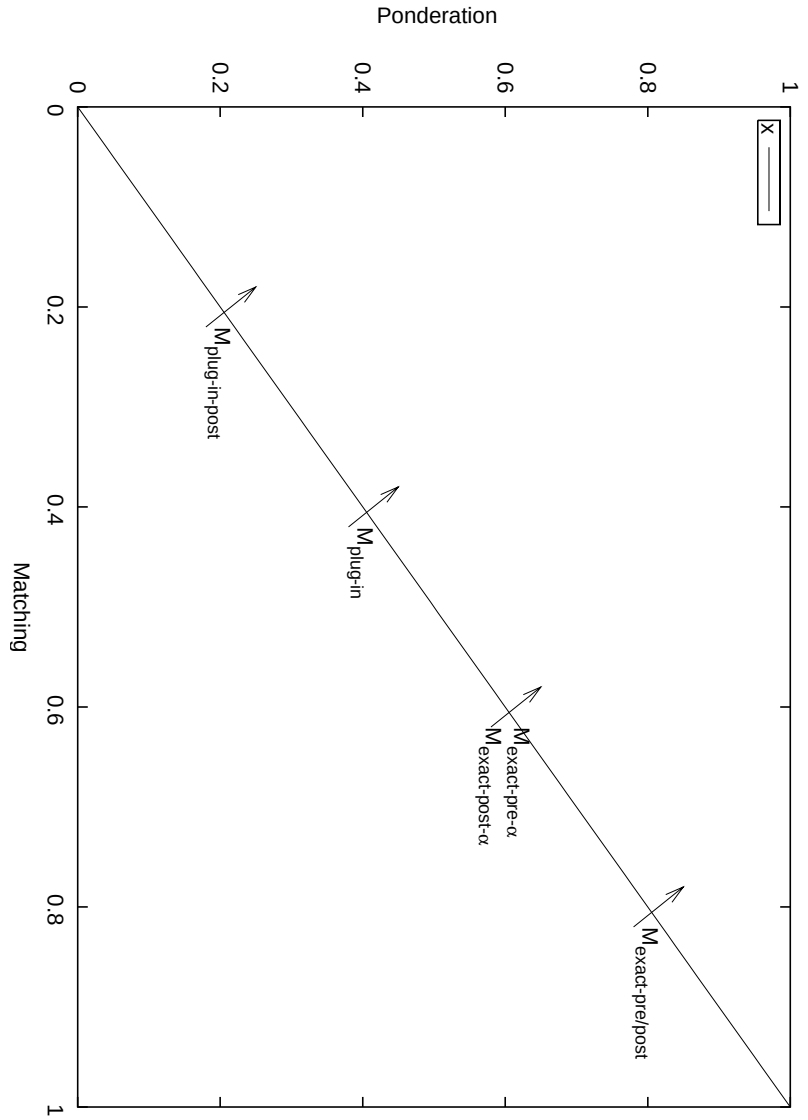


Figure D.5: Pondération des fonctions de Matching de $E \cup F$.

Nous définissons le degré de similitude entre deux services Ws_1 et Ws_2 par la fonction $Sim_S(Ws_1,\ Ws_2)$. Cette fonction permet de déterminer quelle fonction de Matching utilisée

ainsi que sa valeur.

Définition 5. $Sim_S(Ws_1,\ Ws_2) = arg_{Max}(f_p(M))$

Propriété 3. *(Nullité de Sim_S)*
$Sim_S(Ws_1(p_{a_1},\ p_{a_2}),\ Ws_2(p_{b_1},\ p_{b_2})) = 0$ *si et seulement si* $p_{b_2} \nrightarrow p_{a_2}$.

Proof. Traitons les deux implications afin de montrer l'équivalence de la propriété 3.

($\rightarrow$) Supposons l'existence de deux services Ws_1 et Ws_2 définis par $Ws_1(p_{a_1},\ p_{a_2})$ et $Ws_2(p_{b_1},\ p_{b_2})$ avec $p_{a_1} \in Pre(Ws_1), p_{b_1} \in Pre(Ws_2), p_{a_2} \in Post(Ws_1), p_{b_2} \in Post(Ws_2)$. Nous avons l'égalité suivante $Sim_S(Ws_1(p_{a_1},\ p_{a_2}),\ Ws_2(p_{b_1},\ p_{b_2})) = 0$ impliquant $arg_{Max}(f_p(M)) = 0$ d'après la définition 3. Cependant $arg_{Max}(f_p(M))$ est évaluée à nulle si et seulement si $M \notin \{M_{exact-pre/post},\ M_{exact-post-\alpha},\ M_{exact-pre-\alpha}\ M_{plug-in}\ M_{plug-in-post}\}$ et en particulier M ne satisfait pas les conditions de $M_{plug-in-post}$ (la fonction de Matching la moins contraignante). Donc d'après la définition de $M_{plug-in-post}$, la relation $p_{b_2} \rightarrow p_{a_2}$ n'est pas satisfaite donc $p_{b_2} \nrightarrow p_{a_2}$.

($\leftarrow$) Supposons l'existence de deux services Ws_1 et Ws_2 définis par $Ws_1(p_{a_1},\ p_{a_2})$ et $Ws_2(p_{b_1},\ p_{b_2})$ avec $p_{a_1} \in Pre(Ws_1), p_{b_1} \in Pre(Ws_2), p_{a_2} \in Post(Ws_1), p_{b_2} \in Post(Ws_2)$. Nous avons la relation suivante $p_{b_2} \nrightarrow p_{a_2}$. Afin d'évaluer $Sim_S(Ws_1(p_{a_1},\ p_{a_2}),\ Ws_2(p_{b_1},\ p_{b_2}))$, nous devons évaluer $arg_{Max}(f_p(M))$ d'après la définition 3. Nous pouvons remarquer d'après le tableau D.1 qu'aucune fonction de Matching ne satisfait la relation $p_{b_2} \nrightarrow p_{a_2}$. En effet l'équivalence satisfait l'implication, et donc ne satisfait pas la non implication. De même l'implication ne satisfait pas la non implication. Ainsi d'après la définition de f_p, $f_p(M) = 0$. Donc d'après la définition 3, $Sim_S(Ws_1(p_{a_1},\ p_{a_2}),\ Ws_2(p_{b_1},\ p_{b_2})) = 0$.

□

D.4.5 Exemple simple

Supposons un service Ws' défini par:

- Ws'(pre *Cinema*, post *Prix*).

qui en fonction d'un cinéma (nom, adresse, identifiant unique...), nous retourne les prix que celui-ci pratique. Ainsi les pré-conditions de ce service renseigne sur le *Cinéma* et le post-conditions nous renseignerons sur les *Prix* en vigueur au sein de ce même cinéma. Dans le but de connaître les informations sur les prix en vigueur pratiqués par un cinéma donné, il est donc nécessaire de connaître le *Cinéma* dont on veut connaître les prix. Afin de connaître cette dernière information, il est nécessaire de composer Ws' avec un autre service dans le but de connaître le *Cinema* en question. Ainsi en utilisant le formalisme introduit en section D.3.1 page 228, nous construisons un service d'aide à la composition noté $S_{AIC}^{Cinema}(Ws')$.

Ainsi $S_{AIC}^{Cinema}(Ws') = Inverse_S^{Cinema}(Ws')$ par définition. Donc $S_{AIC}^{Cinema}(Ws')$ a pour effet les pré-conditions de Ws', et des pré-conditions que nous initialisons à $\emptyset$. Ainsi $S_{AIC}^{Cinema}(Ws')$ est défini par:

$$S_{AIC}(Ws') : \begin{cases} Input \rightarrow Output \\ \emptyset \rightarrow^{p_s} Cinema \end{cases}$$

Dans le but de composer le service Ws', nous cherchons un service Q ayant un maximum de correspondance avec $S_{AIC}^{Cinema}(Ws')$.

Supposons deux services distincts tels que:

- Ws_1(pre Location, post DistrictCinema);
- Ws_2(pre Location, post EntertainementPlace).

Nous recherchons le service Ws_i ayant le plus de similitudes avec $S_{AIC}^{Cinema}(Ws')$. Afin de déterminer le degré de similitude, nous utilisons la fonction $Sim_S(Ws_1,\ Ws_2)$ qui renvoie une valeur de Matching. Ainsi:

- $Sim_S(S_{AIC}^{Cinema}(Ws'),\ Ws_1) = 0.4$ car $S_{AIC}(Ws')$ et Ws_1 ont une valeur de vérité "Vraie" avec la fonction de correspondance $M_{plug-in}$;
- $Sim_S(S_{AIC}^{Cinema}(Ws'),\ Ws_2) = 0$ car aucune fonction de Matching ne peut s'appliquer à $S_{AIC}^{Cinema}(Ws')$ et Ws_2.

Nous pouvons donc conclure que $S_{AIC}^{Cinema}(Ws')$ et Ws_1 matchent avec une valeur maximale et donc par conséquence que Ws' est composable avec Ws_1 par la fonction de composition:

$$Ws' o Ws_1$$

D.5 Synthèse

Nous avons défini dans ce chapitre un ordre partiel sur les fonctions de Matching. Il était essentiel d'ordonner ces fonctions afin d'affiner la composition automatiques de services web et de permettre un choix fonctionnel de services lorsque celui-ci se présente.

Nous basons notre modèle de sélection de services sur deux paramètres:

- une métrique de Matching inspirée du domaine de l'ingénierie logicielle (Matching de composants logiciels) (Chapitre D);
- un calcul de distance entre concepts (Chapitre C).

Appendix E

Fonction de Similarité Sim_T

E.1 Introduction

Dans ce chapitre nous présentons la fonction de calcul de similarité notée Sim_T, permettant de pondérer la similitude de deux services web Ws et Ws_k. Cette fonction détermine tout d'abord la similarité des services au moyen de la fonction Sim_S, puis détermine la similarité des concepts représentant les pré-conditions de Ws (resp. post-conditions) et les concepts représentant les pré-conditions de Ws_k (resp. post-conditions). Cette dernière similarité est évaluée au moyen de la fonction Sim_C (et donc au moyen de $Sim_{\supseteq}$).

E.2 Fonction de calcul de similarité: Sim_T

E.2.1 Définition

Étant données deux fonctions de similarité Sim_S (Calcul de similarité de deux services) et Sim_C (Calcul de similarité de concepts), nous introduisons Sim_T avec la définition suivante.

Définition 6. *(Fonction de calcul de similarité: Sim_T)*
Si $Sim_S(Ws(p_a,\ p_b),\ Ws_k(p_c,\ p_d)) > 0$ *alors*

$$Sim_T(Ws(p_a,\ p_b),\ Ws_k(p_c,\ p_d)) = Sim_S(Ws(p_a,\ p_b),\ Ws_k(p_c,\ p_d)) + Sim_C(Ws(p_a,\ p_b),\ Ws_k(p_c,\ p_d)) \quad \text{(E.1)}$$

Sinon $Sim_T(Ws(p_a,\ p_b),\ Ws_k(p_c,\ p_d)) = 0$

Nous définissons donc la fonction de calcul de similarité Sim_T comme un application linéaire de Sim_S et Sim_C où la fonction Sim_C est définie par:

$$Sim_C(Ws(p_a,\ p_b),\ Ws_k(p_c,\ p_d)) = \frac{1}{10}\Big(\frac{1}{\#(p_{d_i} \subseteq p_{b_j})} \sum_{p_{d_i} \subseteq p_{b_j}} Sim_{\supseteq}(p_{d_i},\ p_{b_j}) + \frac{1}{\#(p_{a_i} \subseteq p_{c_j})} \sum_{p_{a_i} \subseteq p_{c_j}} Sim_{\supseteq}(p_{a_i},\ p_{c_j})\Big) \quad \text{(E.2)}$$

La fonction Sim_C définit la similitude conceptuelle des pré-conditions de Ws et de Ws_k, ainsi que celles des post-conditions de Ws et de Ws_k. De plus Sim_C se base sur la fonction $Sim_{\supseteq}$ définie en section C.4.1 page 221.

Le calcul de similarité Sim_T dépend donc de la similitude des deux services web Ws et Ws_k, ainsi que de la proximité des paramètres de chacun des services web Ws et Ws_k.

E.2.2 Propriétés essentielles

Propriété 4. *(Domaine image de Sim_C)*
La fonction de similarité Sim_C est une application à valeurs dans l'intervalle $[0,\ \frac{1}{5}]$.

Proof. Par définition $Sim_{\supseteq}$ est à valeurs dans $[0,\ 1]$. En effet $Sim_{\supseteq}(p_{a_i},\ p_{c_j})$ a pour valeur maximale 1 lorsque p_{a_i} est sémantiquement équivalent à p_{c_j}, et pour valeur minimale 0 lorsque p_{a_i} et p_{c_j} ne sont pas comparables par l'opérateur de subsomption. On a donc $Sim_{\supseteq}(p_{a_i},\ p_{c_j}) \in [0,\ 1]\ \forall p_{a_i},\ p_{c_j} \in \mathcal{T}$.

Ainsi nous avons la relation suivante:

$0 \leq Sim_{\supseteq}(p_{a_i},\ p_{c_j}) \leq 1$
$\Leftrightarrow 0 \leq \sum_{p_{a_i} \subseteq p_{c_j}} Sim_{\supseteq}(p_{a_i},\ p_{c_j}) \leq \#\{p_{a_i} \subseteq p_{c_j}\}$
$\Leftrightarrow 0 \leq \frac{1}{\#\{p_{a_i} \subseteq p_{c_j}\}} \sum_{p_{a_i} \subseteq p_{c_j}} Sim_{\supseteq}(p_{a_i},\ p_{c_j}) \leq 1$
$\Leftrightarrow 0 \leq \frac{1}{10}\left(\frac{1}{\#\{p_{a_i} \subseteq p_{c_j}\}} \sum_{p_{a_i} \subseteq p_{c_j}} Sim_{\supseteq}(p_{a_i},\ p_{c_j}) \right) \leq \frac{1}{10}$

Sim_C est une fonction composée d'un produit de deux fractions du même type que ci-dessus. Ainsi nous déduisons que le domaine image de Sim_C est $[0,\ \frac{1}{5}]$. □

Propriété 5. *(Domaine image de Sim_T)*
La fonction de similarité Sim_T est une application à valeurs dans l'intervalle $[0,\ 1]$.

Proof. Par définition Sim_S est à valeurs dans $\mathcal{I}m_{Sim_S} = \{0,\ 0.2,\ 0.4,\ 0.6,\ 0.8\}$. De plus d'après la propriété 4, Sim_C est à valeurs dans $[0,\ \frac{1}{5}]$. Ainsi le domaine image de Sim_T est défini par:

$$\mathcal{I}m_{Sim_T} = [min(\mathcal{I}m_{Sim_S}) + min(\mathcal{I}m_{Sim_C}),\ max(\mathcal{I}m_{Sim_S}) + max(\mathcal{I}m_{Sim_C})]$$

puisque $Sim_T = Sim_S + Sim_C$. D'où $\mathcal{D}_{Sim_T} = [0,\ 1]$. □

Sim_T est évaluée à zero si et seulement si $Sim_S = 0$. En effet la similarité finale Sim_T n'a pas de sens si $Sim_S = 0$, puisque cela signifie $\#\{p_{a_i} \subseteq p_{c_j}\} = 0$ et $\#\{p_{d_i} \subseteq p_{b_j}\} = 0$. Ainsi Sim_C n'est pas calculable puisque le calcul de Sim_C implique un calcul d'un rapport avec dénominateur nul. C'est pourquoi nous définissons Sim_T selon la définition 6.

Propriété 6. *(Validité de Sim_T)*
$Sim_T(Ws(p_a,\ p_b),\ Ws_k(p_c,\ p_d)) = 0$ *si et seulement si* $p_{a_i} \nrightarrow p_{c_j}$

Proof. Traitons les deux implications afin de montrer l'équivalence de la propriété 6.

(→) Supposons l'existence de deux services Ws_1 et Ws_2 définis par $Ws_1(p_a,\ p_b)$ et $Ws_2(p_c,\ p_d)$ avec $p_a \in Pre(Ws_1)$, $p_c \in Pre(Ws_2)$, $p_b \in Post(Ws_1)$, $p_d \in Post(Ws_2)$. Nous avons l'égalité suivante $Sim_T(Ws_1(p_a,\ p_b),\ Ws_2(p_c,\ p_d)) = 0$, alors d'après la définition 6, $Sim_S(Ws(p_a,\ p_b),\ Ws_k(p_c,\ p_d)) = 0$. Nous avons cette dernière égalité si et seulement si $p_{b_2} \nrightarrow p_{a_2}$ d'après la propriété *"Nullité de Sim_S"*.

(←) Supposons l'existence de deux services Ws_1 et Ws_2 définis par $Ws_1(p_a,\ p_b)$ et $Ws_2(p_c,\ p_d)$ avec $p_a \in Pre(Ws_1)$, $p_b \in Pre(Ws_2)$, $p_c \in Post(Ws_1)$, $p_d \in Post(Ws_2)$. Nous avons la relation suivante $p_a \nrightarrow p_b$. Afin de calculer Sim_T, il est nécessaire de connaître au préalable Sim_S, comme l'exige la définition 6. Or d'après la propriété *"Nullité de Sim_S"*, $Sim_S(Ws(p_a,\ p_b),\ Ws_k(p_c,\ p_d)) = 0$ puisque $p_a \nrightarrow p_b$. Donc d'après la définition 6, $Sim_T(Ws(p_a,\ p_b),\ Ws_k(p_c,\ p_d)) = 0$.

□

E.3 Exemple

Dans le but d'illustrer notre fonction de similarité Sim_S, nous présentons un exemple. Dans l'exemple suivant, nous supposons une ontologie unique. Le domaine d'application concerné est le cinéma. Afin de simplifier la compréhension de l'exemple, nous présentons les trois taxonomies utilisées (Figure E.1) issues de l'ontologie relative au domaine du cinéma.

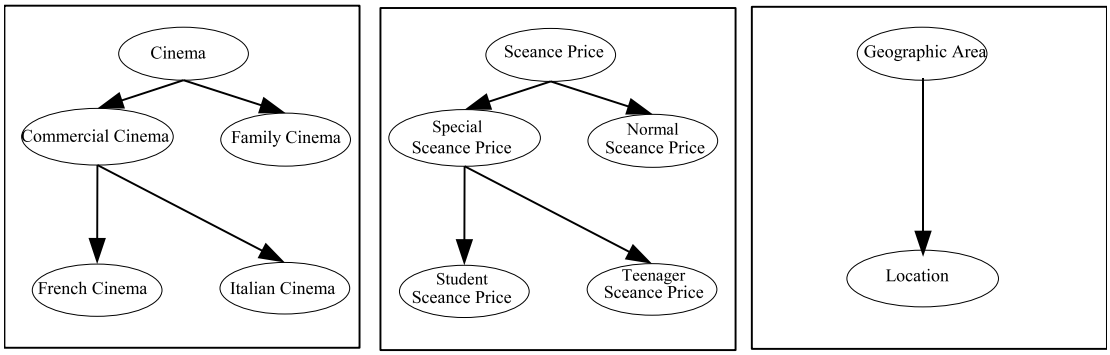


Figure E.1: Taxonomie portant sur le Cinéma.

De plus nous présentons les six concepts décrits en Logique de description $\mathcal{ALN}$ illustrés en figure E.2.

Nous avons décrit le domaine dans lequel nous allons travailler. Nous allons maintenant détailler la fonction de similarité Sim_T au moyen d'un exemple.

Exemple

Tout d'abord supposons deux services web Ws_1 et Ws_2 décrits au moyen de la connaissance de leurs pré-conditions et effets. Ainsi Ws_1 et Ws_2 sont décrits par:

- Ws_1(in FrenchCinema, in Movie, in Date, out Price, out GeographicArea);
- Ws_2(in Cinema, out SpecialPrice, out Location, out OpeningHour).

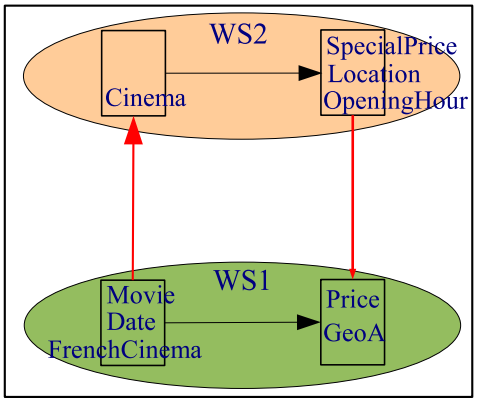
Nous nous apprêtons à calculer la similarité des deux services au sens de notre fonction de similarité Sim_T. D'après la définition 6 (voir (E.1)), Sim_T est vue comme la somme de deux composantes Sim_S et Sim_C. Cependant la valeur de Sim_C est déterminée par la connaissance de Sim_S (toujours d'après la définition 6). Ainsi nous proposons de calculer la fonction Sim_S grâce à la connaissance de Ws_1 et Ws_2.

Match(Ws_1, Ws_2)	Valeur de vérité	Poids $f_p(M)$
1.$M_{exact-pre/post}(Ws_1, Ws_2)$	False	0.8
2.$M_{exact-pre-\alpha}(Ws_1, Ws_2)$	False	0.6
3.$M_{exact-post-\alpha}(Ws_1, Ws_2)$	False	0.6
4.$M_{plug-in}(Ws_1, Ws_2)$	True	0.4
5.$M_{plug-in-post}(Ws_1, Ws_2)$	True	0.2

Table E.1: $f_p(M)$ pour Ws_1 et Ws_2.

$$
\begin{aligned}
&Cinema \equiv (\geq 1\ hasAddress) \cap (\geq 1\ hasLocation) \cap (\geq 1\ hasName) \\
&\qquad \cap (\geq 1\ hasSceancePrice) \\
&FrenchCinema \equiv (\geq 1\ hasAddress) \cap (\geq 1\ hasLocation) \\
&\qquad \cap (\geq 1\ hasName) \cap (\forall\ playMovies.French) \\
&\qquad \cap (\geq 1\ hasSceancePrice) \\
&\qquad \cap (\geq 15 hasAdvertisingContract) \\
&SeancePrice \equiv (= 1\ hasName) \cap (= 1\ hasValue.Int) \\
&\qquad \cap (\geq 1\ belongs.Cinema) \\
&StudentSeancePrice \equiv (= 1\ hasName) \cap (= 1\ hasValue.Int) \\
&\qquad \cap (\geq 1\ belongs.Cinema) \cap (\forall\ isCustomer.Student) \\
&GeographicArea \equiv (= 1\ hasName) \cap (= 1\ isInCountry) \\
&Location \equiv (= 1\ hasName) \cap (= 1\ isInCountry) \\
&\qquad \cap (= 1\ hasAltitude) \cap (= 1\ hasLatitude) \cap (= 1\ hasLongitude)
\end{aligned}
$$

Figure E.2: Exemple d'une Terminologie $\mathcal{T}$ portant sur le Cinéma.



1. Calcul de Sim_S
   D'après la définition 5 du chapitre 7, $Sim_S(Ws_1,\ Ws_2) = arg_{Max}(f_p(M))$ avec M la fonction de Matching appropriée. Les résultats de $f_p(M)$ sont illustrés dans le tableau E.1. Si l'on se réfère au tableau, nous concluons aisément que

   $$Sim_S(Ws_1,\ Ws_2) = 0.4$$

   qui est une valeur strictement positive.

Figure E.3: Matching entre Ws_1 et Ws_2.

Explicitons le tableau E.1. La fonction $M_{exact-pre/post}$ possède une valeur de vérité "False" puisque les pré-conditions de Ws_1 et Ws_2 ne sont pas équivalentes (il en est de même pour les post-conditions). La fonction $M_{exact-pre-\alpha}$ (resp $M_{exact-post-\alpha}$) possède une valeur de vérité "False" puisque les pré-conditions (resp. post-conditions) de Ws_1 et Ws_2 ne sont pas équivalentes. La fonction $M_{plug-in}$ possède une valeur de vérité "True" puisque les pré-conditions de Ws_1 sont plus spécifiques que les pré-conditions de Ws_2 (i.e. les pré-conditions de Ws_1 impliquent les pré-conditions de Ws_2). De plus les post-conditions de Ws_2 sont plus spécifiques que les post-conditions de Ws_1. La fonction $M_{plug-in-post}$ possède une valeur de vérité "True" puisque la fonction $M_{plug-in}$ possède une valeur de

vérité "True" (implication logique dû à la proposition d'ordre total et partiel sur les fonctions de Matching).

2. Calcul de Sim_C

   D'après la définition de Sim_C définie en (E.2), le calcul de Sim_C s'obtient via la connaissance de la fonction de similarité des concepts $Sim_{\supseteq}$. Ainsi nous introduisons le tableau E.2 renseignant sur la similitude des concepts. Notons que la définition de Sim_C introduit des comparaisons de concepts p_{a_i} subsumé par p_{c_j}. Nous supposons que le calcul de $Sim_{\supseteq}$ est effectué avec les valeurs $(\alpha,\ \beta) = (\frac{1}{2},\ \frac{1}{2})$. Ainsi d'après la connaissance de $Sim_{\supseteq}$, de la description des concepts, ainsi que de la taxonomie, nous obtenons les résultats $Sim_{\supseteq}(p_{a_i}, p_{c_j})$ avec p_{a_i} faisant référence à une pré-condition de Ws_1 (resp. post-condition de Ws_2) et p_{c_j} faisant référence à une pré-condition de Ws_2 (resp. post-condition de Ws_1).

$Sim_{\supseteq}(p_x,\ p_y)$	$(\alpha,\ \beta) = (\frac{1}{2},\ \frac{1}{2})$		
	Cinema	SceancePrice	GeoArea
FrenchCinema	0.33	N/A	N/A
SpecialSceancePrice	N/A	0.41	N/A
Location	N/A	N/A	0.37

Table E.2: $Sim_{\supseteq}$ portant sur les pré-conditions et effets de Ws_1 et Ws_2.

   En décomposant le formalisme introduit en (E.2) pour définir Sim_C, nous obtenons respectivement les deux sommes sur les pré-conditions et effets suivantes:
   $\sum_{p_{d_i} \subseteq p_{b_j}} Sim_{\supseteq}(p_{d_i},\ p_{b_j}) = 0.41 + 0.37$ avec $p_{d_i} \in \text{Post}(Ws_2)$ et $p_{b_j} \in \text{Post}(Ws_1)$;
   $\sum_{p_{a_i} \subseteq p_{c_j}} Sim_{\supseteq}(p_{a_i},\ p_{c_j}) = 0.33$ avec $p_{a_i} \in \text{Pre}(Ws_1)$ et $p_{c_j} \in \text{Pre}(Ws_2)$;
   Ainsi d'après (E.2)

$$Sim_C = \frac{1}{10}(\frac{1}{2}(0.41 + 0.37) + 0.37) = 0.076$$

3. Calcul de Sim_T

   D'après la définition 6, $Sim_T = Sim_S + Sim_C = 0.476$.

4. Remarques

   Notons que $Sim_T(Ws_2,\ Ws_1) = 0$ puisque d'après la définition 6 si $Sim_S(Ws_2,\ Ws_1) = 0$ alors $Sim_T(Ws_2,\ Ws_1) = 0$. Nous remarquons donc que notre fonction de similarité Sim_T n'est pas symétrique. Il est trivial de montrer que la fonction de similarité Sim_T n'est pas symétrique. En effet Sim_T, est composée de deux fonctions Sim_C et Sim_S qui sont toutes deux non symétriques.

E.4 Synthèse

Ce chapitre nous a permis de définir la fonction de similarité notée Sim_T, permettant d'évaluer la similarité de deux services Ws_1 et Ws_2 en prenant en compte la similitude des concepts de leurs pré-conditions et effets. Cette fonction de similitude est à la base de notre modèle de composition, puisqu'à partir de celle-ci et de la définition de $S^{it}_{AIC}(S)$ (définition 4 chapitre D),

nous pouvons effectuer un choix fonctionnel en cas de multiples compositions possibles (i.e si un service peut être composé avec plusieurs autres services). Le chapitre suivant introduira la matrice de Matching (ou Matrice de liens sémantiques), élément de base pour la résolution d'un problème de planification que nous définirons lors du chapitre G.

Appendix F

Construction de la matrice de liens sémantiques

F.1 Introduction

Dans ce chapitre, nous présenterons et introduirons la matrice de Matching $\mathscr{M}$ permettant d'aider au processus de composition automatisée de web services. De plus nous présenterons deux algorithmes permettant la construction d'une telle matrice.

F.2 Matrice de Matching (ou liens sémantiques) $\mathscr{M}$

Étant donnés une fonction de similitude Sim_T sur les web services à paramètres dans une ontologie $\mathcal{T}$ et un ordre total sur les fonctions de Matching, nous introduisons l'algorithme de construction de la matrice $\mathscr{M}$ à double entrée. Au préalable, nous définirons formellement la matrice de Matching notée $\mathscr{M}$.

Définition 7. *(Matrice de Matching)*
Soit $M_{p,q}(S_{WebServices} * [0,\ 1])^l$ l'espace des matrices $p \times q$ à coefficients dans $(S_{WebServices} * [0,\ 1])^l$. Une Matrice de Matching $M \in M_{p,q}(S_{WebServices} * [0,\ 1])^l$ représente une matrice à plusieurs entrées. Chaque entrée est assimilée à un ensemble de couples $(W_s,\ score) \in S_{WebServices} * [0,\ 1]$ où W_s représente un service web découvert et disponible, et score sera analysé comme un coefficient de Matching, explicité par:

$$score = Sim_T(s(\mathcal{KB},\ j),\ \mathcal{M}(i,\ j).s_x) \text{ avec } i \in \{1, ..., p\} \text{ et } j \in \{1, ..., q\}$$

Nous définissons donc $\mathscr{M}$ comme à une matrice à multiples entrées.

Exemple 1.
Si l'on considère la matrice $M_1 \in M_{2,2}(S_{WebServices} * [0,\ 1])$ suivante:

$$M_1 = \begin{pmatrix} \{(\emptyset, 0)\} & \{(S_1, 0.5)\} \\ \{(S_2, 0.8)\} & \{(\emptyset, 0)\} \end{pmatrix}$$

Nous analysons M_1 comme un matrice à coefficients dans $S_{WebServices} * [0,\ 1]$ où $M_1(1,\ 1)$ et $M_1(2,\ 2)$ ont un score nul et un ensemble de services web correspondant à l'ensemble vide. La

valeur de $M_1(1,\ 1)$ signifie qu'il n'existe pas de services web sémantiquement comparables[1] à un service web ayant des pré-conditions p_1 et effets e_1. De même, la valeur de $M_1(2,\ 2)$ signifie qu'il n'existe pas de services web sémantiquement comparables à un service ayant des pré-conditions p_2 et effets e_2. Contrairement aux valeurs de $M_1(1,\ 1)$ et $M_1(2,\ 2)$, $M_1(1,\ 2)$ et $M_1(2,\ 1)$ ont un score non nul et un ensemble de services non vide. Cela signifie qu'il existe un service web S_1 (resp. S_2) sémantiquement comparable à un service web ayant des pré-conditions p_1 (resp. p_2) et effets e_2 (resp. e_1).

Nous introduisons formellement la Matrice de Matching nulle pour des raisons requises par les algorithmes présentés en section suivante.

Définition 8. *(Matrice de Matching nulle)*
Une Matrice de Matching $M \in M_{p,q}(S_{WebServices} * [0,\ 1])^l$ *est nulle si et seulement si*

$$M(i,j) = \{(\emptyset,\ 0)\} \qquad \forall i \in \{1,...,p\},\ \forall j \in \{1,...,q\}$$

Cette matrice sera notée $M_\emptyset$.

La Matrice de Matching nulle permettra d'initialiser l'environnement.

Définition 9. *(Matrice de Matching non nulle)*
Une Matrice de Matching $M \in M_{p,q}(S_{WebServices} * [0,\ 1])^l$ *est non nulle si et seulement si*

$$\exists i \in \{1,...,p\},\ \exists j \in \{1,...,q\} \text{ tel que } M(i,j) \neq \{(\emptyset,\ 0)\}$$

Définition 10. *(Éléments non nuls de la matrice de Matching)*
Un élément $\mathscr{M}(i,\ j)$ *de la matrice de Matching* $\mathscr{M}$ *est non nul si et seulement si* $\mathscr{M}(i,\ j) \neq \{(\emptyset,\ 0)\}$.

Propriété 7. *Une matrice de Matching* $\mathscr{M} \in M_{p,q}(S_{WebServices} * [0,\ 1])^l$ *est non nulle si et seulement s'il existe un élément non nul de la matrice de Matching* $\mathscr{M}$.

Proof. Traitons les deux implications afin de montrer l'équivalence de la propriété 7.

(←) Supposons qu'il existe un élément non nul de la matrice de Matching $\mathscr{M}$. Alors d'après la définition 10, il existe $i \in \{1,...,p\}$ et $j \in \{1,...,q\}$ de sorte que $\mathscr{M}(i,\ j) \neq \{(\emptyset,\ 0)\}$. Donc d'après la définition 9, la matrice de Matching $\mathscr{M} \in M_{p,q}(S_{WebServices} * [0,\ 1])^l$ est non nulle.

(→) Supposons que $\mathscr{M} \in M_{p,q}(S_{WebServices} * [0,\ 1])^l$ soit non nulle, alors d'après la définition 10, il existe donc un élément non nul de la matrice de Matching $\mathscr{M}$.

□

F.3 Construction de la matrice $\mathscr{M}$ liant les pré-conditions

L'algorithme 6 et 7 (resp. 6 et 8) présentent le processus de construction de la Matrice de Matching. L'algorithme 6 a pour but de créer et initialiser les éléments de la liste $\mathcal{L}$ (nécessaire pour l'algorithme 8) et la matrice $\mathscr{M}$ alors que l'algorithme 7 construit la matrice $\mathscr{M}$ pas à pas. L'algorithme 7 introduit le processus naïf de construction de $\mathscr{M}$, alors que l'algorithme 8 fournit un processus optimisé de construction de $\mathscr{M}$.

[1] La comparaison sémantique se fait à l'aide de la fonction de similitude Sim_T.

F.3.1 Idée générale

L'idée générale de la construction d'une matrice $\mathscr{M}$ est d'obtenir un matrice à p lignes[2] et q colonnes[3] permettant d'aider à la construction d'un plan satisfaisant un but précis. Pour satisfaire une telle ambition, il est nécessaire d'avoir connaissance d'un ensemble d'ontologies $\mathcal{T}$, d'un ensemble de services $S_{WebServices}$, d'un ensemble de buts locaux β_L, ainsi qu'une fonction de similitude Sim_T définie lors du chapitre E. L'ensemble des ontologies $\mathcal{T}$ permet la comparaison des concepts décrits en Logique de description. L'ensemble des services permettra la construction d'un plan de services. L'ensemble des buts locaux indique la direction à suivre pour résoudre le but final et global. Enfin la fonction de similitude permet de comparer un service abstrait à pré-condition et post-condition unique avec un service appartenant à $S_{WebServices}$ dans le but d'obtenir un indice reflétant le degré de similitude des deux services comparés. Ainsi un service S de $S_{WebServices}$ sera composable avec un service S' de $S_{WebServices}$ si $S \subseteq \mathcal{M}(p_a,\ p_b).set$ avec $p_b \in PreCondition(S')$.

F.3.2 Algorithme de création et d'initialisation

Algorithm 6: Instanciation des paramètres nécessaires à la construction de la Matrice $\mathscr{M}$.

Données: un ensemble de services Web: $S_{WebServices} = \{Ws_1, ..., Ws_n\}$,
un ensemble d'ontologies: $T = \{T_1, ..., T_m\}$,
l'ensemble des buts locaux $\beta_L = \{b_{l_1}, ..., b_{l_p}\}$.

Résultat: $\mathcal{L}$ instanciée et initialisée,
$\mathscr{M}$ instanciée et initialisée.

début
  //Initialisation de la liste $\mathcal{L}$
  $\mathcal{L} \leftarrow \emptyset$
  pour chaque *but local* $b_i \in \beta_L$ **faire**
    Ajouter(b_i, $\mathcal{L}$);
  fin
  //Initialisation de la matrice $\mathscr{M}$
  pour chaque *pré-condition* p_a *distincte de* $\bigcup_{i=1}^{n} pre(Ws_i)$ **faire**
    pour chaque *pré-condition* p_b *distincte de* $\bigcup_{i=1}^{n} pre(Ws_i) \cup \beta_L$ **faire**
      $\mathscr{M}(p_a,\ p_b) \leftarrow \{(\emptyset,\ 0)\}$;
    fin
  fin
fin

1. **Preuve d'arrêt:**

   L'arrêt de la première boucle (*Initialisation de la liste* $\mathcal{L}$) de l'algorithme 6 est assuré puisque β_L est un ensemble fini de buts locaux. Par conséquent, l'analyse d'un ensemble fini s'arrête en temps fini.

   L'ensemble des services web appartenant à $S_{WebServices}$ et l'ensemble des buts locaux β_L sont finis par construction. De plus l'ensemble des pré-conditions et effets de chaque service

[2] $p = Card(Preconditions(S_{WebServices}))$.
[3] $q = p + Card(\beta_l) - Card(\beta_l \cap\ Preconditions(S_{WebServices}))$.

web est fini par définition. En effet chaque service web est défini comme une entité présente sur le web avec un ensemble fini de pré-conditions et effets. Ainsi l'arrêt de la boucle portant sur l'*Initialisation de la matrice* $\mathscr{M}$ est assuré. L'arrêt de l'algorithme 6 est ainsi prouvé.

2. **Preuve de correction:**

   L'objectif principal de l'algorithme 6 est de créer et d' initialiser la liste $\mathcal{L}$ et la matrice $\mathscr{M}$. Ces deux tâches sont effectuées lors des deux premières boucles. En effet après avoir initialisé la liste $\mathcal{L}$ à $\emptyset$, l'algorithme 6 initialise la $\mathcal{L}$ en ajoutant tous les buts locaux $b_i \in \beta_L$ à la liste $\mathcal{L}$. Il en est de même pour la matrice de Matching $\mathscr{M}$ qui est instanciée et initialisée à la matrice de Matching nulle $M_\emptyset$. Notons que $M \in M_{p,q}(S_{WebServices} * [0,\ 1])^l$ où $l = 1$ et p et q sont définis par:

$$p = Card(\bigcup_{i=1}^{n} pre(Ws_i)) \tag{F.1}$$

$$q = Card(\bigcup_{i=1}^{n} pre(Ws_i) \cup \beta_L) \tag{F.2}$$

   La preuve de la correction de l'algorithme 6 est achevée puisque nous avons montré que l'instanciation et l'initialisation de $\mathcal{L}$ et $\mathscr{M}$ sont effectuées correctement. En effet tous les éléments de $\mathscr{M}$ sont initialisés à l'élément vide $\{(\emptyset,\ 0)\}$ du fait du parcours de la matrice ligne par ligne puis colonne par colonne.

3. **Analyse de la complexité:**

   L'initialisation de la liste $\mathcal{L}$ s'effectue en temps linéaire en fonction de la taille de β_L. Ainsi la complexité de l'instanciation de la liste $\mathcal{L}$ est de $\theta(Card(\beta_L))$.

   L'initialisation de la matrice de Matching $\mathscr{M}$ s'effectue en temps quadratique en fonction de p et q. Ainsi la complexité de l'initialisation de $\mathscr{M}$ est de l'ordre de $\theta(pq)$. La complexité du processus d'instanciation et d'initialisation est donc dans le pire des cas quadratique ou $\theta(pq)$.

Notations:

Pour des raisons de lisibilité, nous introduisons quelques notations qui seront utilisées dans les algorithmes suivants. Tout d'abord, chaque matrice de Matching $\mathscr{M}$ sera un élément de $M_{p,q}(S_{WebServices} * [0,\ 1])^l$, où p représente son nombre de ligne et q son nombre de colonne. Ainsi pour chaque matrice de Matching $\mathscr{M}$, nous définissons p (resp. q) par la formule (F.1) en section F.1 page 250 (resp. (F.2) en section F.1 page 250).

F.3.3 Algorithme de construction de $E_s s(W_s(p_a,\ p_b))$

L'ensemble $E_s s(W_s(p_a,\ p_b))$ représente un ensemble permettant de déterminer quels services web seront présents dans la matrice de Matching $\mathcal{M}$ en ligne p_a et colonne p_b. Cet ensemble permet donc de déterminer les liens sémantiques (liens service-service) entre chaque service de $S_{WebServices}$. De plus $E_s s$ ne présentera que des services web ayant une similarité strictement positive avec un service de type $W_s(p_a,\ p_b)$. Nous donnons une définition formelle de $E_s s$ lors de la définition 11.

Définition 11. *(Ensemble de services similaires de W_s)*
Nous définissons l'ensemble des services similaires de W_s noté $E_ss(W_s)$ comme:

$$E_ss(W_s) = \{Ws_k \mid Sim_T(W_s(p_a\ , p_b),\ WS_k) > 0\}$$
$$avec\ p_a \in Precondition(W_s),\ p_b \in Postcondition(W_s)\ et\ W_s \in S_{AIC}. \quad \text{(F.3)}$$

Algorithm 7: Construction de l'ensemble E_ss pour le service $W_s(p_a, p_b)$.

Données: un ensemble de services Web: $S_{WebServices} = \{Ws_1, ..., Ws_n\}$,
un ensemble d'ontologies: $\mathcal{T} = \{T_1, ..., T_m\}$,
un service web $Ws(p_a,\ p_b)$.
Résultat: $E_ss(W_s(p_a,\ p_b))$
début
  $E_ss(W_s(p_a,\ p_b)) \leftarrow \emptyset$;
  pour chaque *service web* $Ws_k \in S_{WebServices}$ **faire**
    si $Sim_T(W_s(p_a\ ,\ p_b), Ws_k) > 0$ **alors**
      Ajouter(Ws_k, $E_ss(W_s(p_a,\ p_b))$);
    fin
  fin
  Retourner $E_ss(W_s(p_a,\ p_b))$;
fin

1. **Preuve d'arrêt:**

   Tout comme l'initialisation de l'ensemble $E_ss(W_s(p_a,\ p_b))$ à $\emptyset$ la fonction *"Ajouter"* est une opération s'effectuant en temps fini. Le parcours de l'ensemble des services web de $S_{WebServices}$ s'effectue en temps fini puisque celui-ci est fini par construction. De plus le test de similarité est une opération déterministe. Nous déduisons donc que l'algorithme 7 s'arrête en temps fini.

2. **Preuve de correction:**

   Si l'on suppose $S_{WebServices} = \emptyset$, alors aucun service similaire n'est retrouvé donc $E_ss(W_s(p_a,\ p_b)) = \emptyset$, ce qui est correct. S'il existe un service Ws_k web appartenant $S_{WebServices}$ tel que $Sim_T(WS(p_a\ ,\ p_b), Ws_k) > 0$, alors celui-ci est ajouté a l'ensemble $E_ss(W_s(p_a,\ p_b))$. Ainsi Ws_k vérifie bien les conditions de l'ensemble des services similaires $E_ss(W_s(p_a,\ p_b))$ (définition 11). Ainsi nous avons prouvé la correction de l'algorithme.

3. **Analyse de la complexité:**

   La complexité de l'algorithme est fonction du nombre de services web présents dans l'ensemble $S_{WebServices}$. Ainsi

   $$Complexite(Algorithme\ 7) = \theta(n)$$

   qui est donc linéaire en nombre de services web de $S_{WebServices}$.

F.3.4 Algorithme naïf de construction de la matrice $\mathscr{M}$

Propriété 8. *(Éléments non nuls de la Matrice de Matching)*
Un élément m de $M \in M_{p,q}(S_{WebServices} * [0,\ 1])^l$ noté $M(i,\ j)$ avec $i \in \{1, ..., p\},\ j \in \{1, ..., q\}$

est non nul si et seulement si

$$\exists s \in S_{WebServices} \textit{ tel que } Sim_T(\mathcal{M}(i,\ j).s_x, s) > 0 \\ \textit{avec } i \subset \ Precondition(s_x) \textit{ et } j \subset \ Postcondition(s_x). \quad \text{(F.4)}$$

Proof. Traitons les deux implications afin de montrer la propriété 8.

(←) Supposons que $\exists s \in S_{WebServices}$ *tel que* $Sim_T(\mathcal{M}(i,\ j).s_x, s) > 0$, alors d'après la définition 11, $Card(\{s \mid Sim_T(\mathcal{M}(i,\ j).s_x,\ s) > 0\}) > 0$, et par conséquent $E_s s(W_s) \neq \emptyset$. Ainsi d'après l'algorithme 8, on itère sur chaque élément de $E_s s(W_s)$, et chaque élément $\mathscr{M}(i,\ j)$ vérifiant $\mathscr{M}(i,\ j) = (\emptyset,\ 0)$ est modifié pour être différent de l'élément nul $(\emptyset,\ 0)$. Ainsi $\mathscr{M}(i,\ j)$ n'est plus assimilé à un élément nul. Si nous nous trouvons dans le cas $\mathscr{M}(i,\ j) \neq (\emptyset,\ 0)$ alors la propriété est vérifiée.

(→) Supposons qu'il existe un élément m de $M \in M_{p,q}(S_{WebServices} * [0,\ 1])^l$ noté $M(i,\ j)$ avec $i \in \{1, ..., p\},\ j \in \{1, ..., q\}$ non nul. Ainsi $M(i,\ j) \neq (\emptyset,\ 0)$. Par conséquent, $\exists s \in S_{WebServices}$ et un réel $r \in \Re \cap [0,\ 1]$ tels que $M(i,\ j) = (s,\ r)$. Nous pouvons donc conclure que $Sim_T(\mathcal{M}(i,\ j).s_x,\ s) = r$ avec $i \in$ Precondition(s_x) et $j \in$ Postcondition(s_x). Ainsi nous avons prouvé l'existence d'un service $s \in S_{WebServices}$ tel que $Sim_T(\mathcal{M}(i,\ j).s_x, s) > 0$.

□

Algorithm 8: Construction naïve de la matrice $\mathscr{M}$ liant les pré-conditions de $S_{WebServices}$.

Données: un ensemble de services Web: $S_{WebServices} = \{Ws_1, ..., Ws_n\}$,
un ensemble d'ontologies: $\mathcal{T} = \{T_1, ..., T_m\}$,
la matrice $\mathscr{M}$ initialisée.

Résultat: la matrice $\mathscr{M}$ liant les pré-conditions des services de $S_{WebServices}$.

début
  pour chaque *pré-condition* p_a *distincte de* $\bigcup_{i=1}^{n} pre(Ws_i)$ **faire**
    pour chaque *pré-condition* p_b *distincte de* $\bigcup_{i=1}^{n} pre(Ws_i) \cup \beta_L$ **faire**
      si $p_b \in \mathcal{KB}$ *tel que* p_b *soit instancié* **alors**
        $\mathscr{M}(p_a,\ p_b) = (\mathcal{KB},\ 1)$;
      sinon
        si $E_s s(W_s(\mathcal{KB},\ p_b)) \neq \emptyset$ **alors**
          pour chaque $Ws_k \in E_s s(W_s(\mathcal{KB},\ p_b))$ **faire**
            si $p_a \in$ *Preconditions*(Ws_k) **alors**
              si $\mathscr{M}(p_a,\ p_b) = (\emptyset,\ 0)$ **alors**
                $\mathscr{M}(p_a,\ p_b) \leftarrow (Ws_k,\ Sim_T(WS(\mathcal{KB},\ p_b),\ Ws_k))$;
              sinon
                $\mathscr{M}(p_a,\ p_b) \leftarrow \mathscr{M}(p_a,\ p_b) \cup (Ws_k,\ Sim_T(WS(\mathcal{KB},\ p_b), Ws_k))$;
              fin
            fin
          fin
        fin
      fin
    fin
  fin
  retourner $\mathscr{M}$;
fin

Analyse de l'algorithme:

Nous introduisons l'algorithme naïf de construction de la matrice $\mathscr{M}$. La matrice $\mathscr{M}$ a pour but d'aider à la construction du plan, et donc du workflow permettant la composition de services web. Le désavantage de ce type de construction est le parcours entier de la matrice $\mathscr{M}$. De plus à chaque itération, nous devons construire et analyser l' ensemble $E_s s(W_s(\mathcal{KB},\ p_b))$ dont le coût de construction est non négligeable. L'algorithme 8 aura pour but d'optimiser la construction de la matrice de Matching $\mathscr{M}$.

1. **Preuve d'arrêt:**

   L'algorithme 8 est décomposé en trois boucles distinctes. La première boucle a pour effet d'itérer sur l'ensemble distinct des pré-conditions p_a de $\bigcup_{i=1}^{n} pre(Ws_i)$. L'ensemble $S_{WebServices}$ est un ensemble fini par définition (i.e. son cardinal est n). De plus le nombre de pré-conditions par services web de $S_{WebServices}$ est fini par définition. Ainsi cette boucle s'effectue en temps fini. La seconde boucle itère sur l'ensemble distinct des pré-conditions p_b de $\bigcup_{i=1}^{n} pre(Ws_i) \cup \beta_L$. Cependant l'ensemble des buts locaux β_L est aussi fini par construction. Comme l'union de deux ensembles finis est fini, nous pouvons conclure que la deuxième boucle s'effectue en temps fini. En outre le test $E_s s(W_s(\mathcal{KB},\ p_b)) \neq \emptyset$ est déterministe. La boucle portant sur les éléments de $E_s s$ s'effectue en temps fini puisque

$E_s s$ est un ensemble fini. Nous pouvons remarquer que toutes les opérations de l'algorithme 8 s'effectuent en temps fini, et par conséquent s'arrêtent. Nous pouvons donc conclure que cet algorithme s'arrête.

2. **Preuve de correction:**

   Cet algorithme a pour but de construire la matrice élément par élément. Trois cas sont envisageables. Le premier laisse un élément $(p_a,\ p_b) \in \mathscr{M}$ à sa valeur initiale $(\emptyset,\ 0)$ si et seulement si $E_s s(W_s(\mathcal{KB},\ p_b)) = \emptyset$ ou $p_a \notin pre(E_s s)$. En effet, il semble correct de n'affecter aucune valeur à un couple $(p_a,\ p_b)$ n'ayant aucun service similaire[4]. Dans le cas contraire, la valeur de $\mathscr{M}(p_a,\ p_b)$ est modifiée afin de renseigner sur l'existence d'un (ou plusieurs) service(s) similaire(s). Enfin si $p_b \in \mathcal{KB}$ alors la valeur de $(p_a,\ p_b) \in \mathscr{M}$ est modifiée pour obtenir $(\mathcal{KB},\ 1)$.

3. **Analyse de la complexité:**

   La complexité de l'algorithme 8 est fonction de deux étapes importantes. L'étape 1 est fonction du nombre pré-conditions p_a distinctes de $\bigcup_{i=1}^{n} pre(Ws_i)$. Supposant p l'ensemble des pré-conditions des services web de $\bigcup_{i=1}^{n} pre(Ws_i)$ et q l'ensemble des pré-conditions p_b distinctes de $\bigcup_{i=1}^{n} pre(Ws_i) \cup \beta_L$, nous avons la relation suivante: $p \leq q$. De plus p, q et $Card(\beta_l)$ sont reliés par la relation suivante:

$$q = p + Card(\beta_l) - Card(\beta_l \cap\ Preconditions(S_{WebServices}))$$

   Soit n le cardinal de $S_{WebServices}$, alors la complexité de l'algorithme 8 est de l'ordre de $\theta(pqn)$, soit dans le pire des cas cubique, en $\theta(q^3)$.

F.3.5 Algorithme optimisé de Construction de la matrice $\mathscr{M}$

L'algorithme 9 optimise l'algorithme précédant. La matrice n'est plus construite élément par élément mais suivant un ordre précis. Cet ordre consiste à remplir la matrice en partant d'un but de β_L et d'itérer jusqu'à obtention d'éléments de $\mathcal{KB}$.

[4]Nous entendons par service similaire, un service web ayant des pré-conditions proche de p_a et des post-conditions proche de p_b (pour simplifier).

Algorithm 9: Construction optimisée de la matrice $\mathscr{M}$ liant les pré-conditions de $S_{WebServices}$.

Données: un ensemble de services Web: $S_{WebServices} = \{Ws_1, ..., Ws_n\}$,
un ensemble d'ontologies: $T = \{T_1, ..., T_m\}$,
la liste $\mathcal{L}$ et la matrice $\mathscr{M}$ initialisées.

Résultat: la matrice $\mathscr{M}$ liant les pré-conditions des services de $S_{WebServices}$.

début
  Copier($\mathcal{L}$, $\mathcal{L}_{temp}$);
  tant que $\mathcal{L} \neq \emptyset$ **faire**
    $e \leftarrow$ Premier($\mathcal{L}$);
    si $e \in \mathcal{KB}$ *tel que e soit instancié* **alors**
      pour chaque *ligne* $l \in \{1, ..., p\}$ *de* $\mathscr{M}$ **faire**
        $\mathscr{M}(l,\ e) = (\mathcal{KB},\ 1)$;
      fin
    sinon
      //$e \in (\beta_L \cup \bigcup_{i=1}^{n} pre(Ws_i) \cup \beta_L$
      Rechercher WS_k *tel que* $Sim_T(WS(\mathcal{KB}\ ,\ e), WS_k) > 0$
      pour chaque WS_k *retrouvé* **faire**
        pour chaque *pré-condition* p_c *distincte de* WS_k **faire**
          si $p_c \notin \mathcal{L}_{temp}$ **alors**
            si $\mathscr{M}(p_c,\ e) = (\emptyset,\ 0)$ **alors**
              $\mathscr{M}(p_c,\ e) \leftarrow (Ws_k, Sim_T(WS(\mathcal{KB}\ ,\ e), Ws_k))$;
            sinon
              $\mathscr{M}(p_c,\ e) \leftarrow \mathscr{M}(p_c,\ e) \cup (Ws_k,\ Sim_T(WS(\mathcal{KB}\ ,\ e), Ws_k))$;
            fin
            Ajouter(p_c, $\mathcal{L}$);
            Ajouter(p_c, $\mathcal{L}_{temp}$);
          fin
        fin
      fin
    fin
    Supprimer(e, $\mathcal{L}$);
  fin
  Retourner $\mathscr{M}$;
fin

Propriété 9. *(Domaine de définition de $\mathcal{L}$)*
$\forall e \in \mathcal{L}$, nous avons une des trois possibilités suivantes:

i) $e \in \mathcal{KB}$;

ii) $e \in \bigcup_{i=1}^{n} pre(Ws_i)$;

iii) $e \in \beta_L$.

Proof. Par construction, la liste $\mathcal{L}$ initiale ne contient que des éléments e appartenant à l'ensemble des buts locaux β_L (voir Algorithme 6). La propriété est donc vraie pour le cas initial.

Supposons que $\exists e \in \mathcal{L}$ et $Ws_k \in S_{WebServices}$ tels que

$$\exists pre_1 \in \bigcup_{i=1}^{n} pre(Ws_i),\ Sim_T(WS(pre_1\ ,\ e), Ws_k) > 0$$

Alors dans ce cas de figure l'algorithme 9 propose d'ajouter à $\mathcal{L}$ l'ensemble des pré-conditions de Ws_k. Ainsi la liste est composée d'éléments de β_L et $Precondition(S_{WebServices})$.

Supposons que $\mathcal{KB} \cap Precondition(S_{WebServices}) \neq \emptyset$, alors $\exists e$ tel que $e \in \mathcal{KB}$ et $e \in Precondition(S_{WebServices})$. □

Propriété 10. *(Non redondance des éléments de $\mathscr{M}$)*
Soit un matrice de matching $M \in M_{p,q}(S_{WebServices} * [0,\ 1])^l$

$$Soit\ m_1,\ m2 \in M(i,\ j),\ alors\ m1 \neq m_2 \qquad \forall i \in \{1, ..., p\},\ \forall j \in \{1, ..., q\}$$

Proof. Raisonnons par l'absurde. Supposons qu'il existe deux éléments m_1 et m_2 de $M_{i,j}$ tels que $m_1 = m_2$. Ainsi il existe un service web noté s_x tel que $\delta = Sim_T(s(i\ ,\ j), s_x) > 0$. On a donc $m_1 = (s_x,\ \delta)$ et $m_2 = (s_x,\ \delta)$. Il est possible d'arriver à cette configuration si et seulement si j a été insérer deux fois (pour un duplicata) dans la liste $\mathcal{L}$ cependant le test:

$$j \notin \mathcal{L}_{temp}$$

permet de ne pas ajouter un élément (ou pré-condition) j si celui-ci a déjà été présent (ou est présent) dans la liste $\mathcal{L}$. La liste $\mathcal{L}_{temp}$ permet donc de contrôler ces cas de disfonctionnement de l'algorithme 9. On a donc prouvé que $m_1 \neq m_2$. □

Analyse de l'algorithme et optimisation:

Afin d'optimiser les performances de l'algorithme de construction de la matrice de Matching $\mathscr{M}$, nous avons introduit un copie de la liste $\mathcal{L}$ notée $\mathcal{L}_{temp}$. Celle-ci est en expansion croissante. En effet, son nombre d'éléments croît au fur et à mesure du déroulement de l'algorithme. Cette liste $\mathcal{L}_{temp}$ tend vers son optimum global lorsque la matrice $\mathscr{M}$ finale est retournée, c'est à dire à la fin de l'exécution de l'algorithme. L'utilité d'une telle liste provient du fait qu'elle renseigne sur les pré-conditions rencontrées au cours de l'exécution de l'algorithme 9 (contrairement à la liste $\mathcal{L}$ qui renseigne seulement sur les pré-conditions à traiter). Cette liste permet donc de ne pas traiter des pré-conditions déjà traitées et qui ne seraient plus présentes dans la liste $\mathcal{L}$. La liste $\mathcal{L}_{temp}$ possède un rôle de control au sein de l'algorithme 9, elle permet donc la non duplication des informations nécessaires à la construction de la matrice de Matching $\mathscr{M}$.

En plus d'obtenir une optimisation, la liste $\mathcal{L}_{temp}$ permet de traiter les cas difficiles de boucles infinies ce qui ne rendraient pas l'algorithme de construction de $\mathscr{M}$ déterministe. Par exemple supposons deux services s_1 et s_2 appartenant à $S_{WebServices}$ tels que:

- s_1(in Location, in i_2,..., in i_n , out Cinema, out o_2,..., o_n);
- s_2(in Cinema, in i'_2,..., in i'_n , out Location, out o'_2,..., o'_n).

Il est évident que s_1 et s_2 sont composables infiniment (voir figure F.1). Ainsi il serait intéressant de traiter ce cas de composition afin d'obtenir un algorithme déterministe pour la composition. La liste $\mathcal{L}_{temp}$ permet d'éliminer le cas des boucles infinies. En effet la liste $\mathcal{L}_{temp}$ a pour but de tester les nouvelles pré-conditions et donc de ne pas traiter les pré-conditions présentes (et donc déjà traitées) dans $\mathcal{L}_{temp}$.

Une telle matrice n'est pas construite itérativement élément par élément, mais récursivement via les éléments utiles de la matrice $\mathscr{M}$.

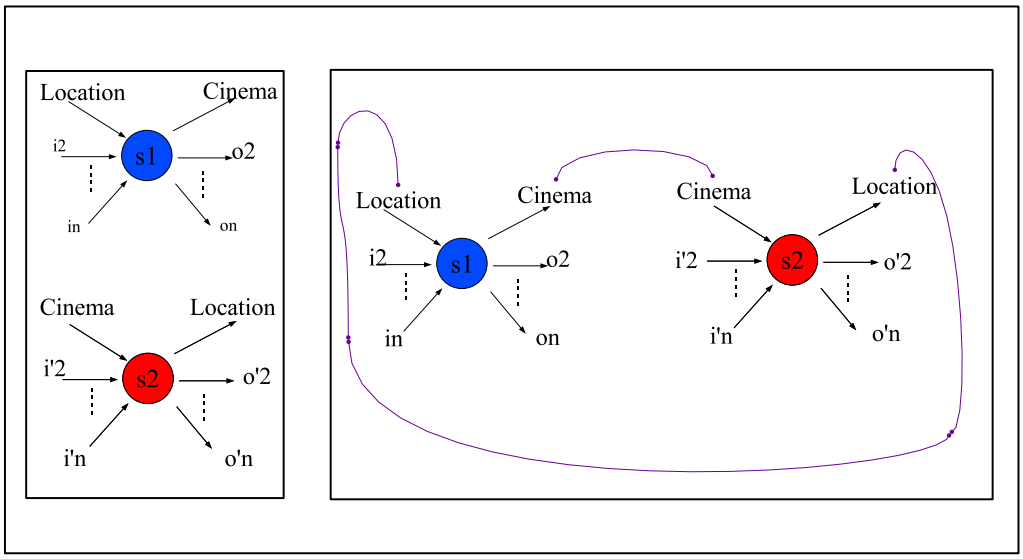


Figure F.1: Cas critique de la composition de services.

1. **Preuve d'arrêt:**

   Afin de prouver l'arrêt de l'algorithme, décomposons l'algorithme en sous parties dans le but d'obtenir un analyse plus fine. Ainsi nous décomposons l'algorithme 9 en trois sous parties indépendantes. Considérons tout d'abord la seconde partie notée *ligne 10*. Cette partie fait référence à une boucle sur un nombre fini de lignes de la matrice $\mathscr{M}$. Par construction, la matrice $\mathscr{M}$ est finie en nombre de lignes et de colonnes, donc cette boucle s'arrête lorsqu'elle atteint la dernière ligne p de la matrice $\mathscr{M}$. Ainsi si $e \in \mathcal{KB}$ alors l'algorithme 9 s'arrête.

   Si l'on suppose que $e \notin \mathcal{KB}$, alors nous nous trouvons dans la partie notée *ligne 16*. Cette partie fait référence à une recherche de services au moyen d'une double boucle. La recherche de services web représentée par *"Rechercher* Ws_k*"* s'effectue en temps fini, et l'arrêt est vérifié du fait de l'aspect déterministe de la recherche de services web. Ainsi le processus de recherche s'effectue en temps fini et s'arrête. Ensuite, nous itérons sur le nombre de services web découverts qui est supposé fini puisque la recherche de services s'effectue sur un ensemble de services web $S_{WebServices}$ de cardinalité n. De plus chaque service web appartenant à $S_{WebServices}$ est fini dans son nombre de paramètres (i.e. pré-conditions et effets). Ainsi l'itération sur le nombre de pré-conditions des services web découverts est un processus fini qui s'arrête. Enfin l'ajout s'effectue en temps fini. Le cas où $e \notin \mathcal{KB}$ est donc un cas s'effectuant en temps fini.

   Traitons maintenant le cas *ligne 7* qui englobe les deux cas traités auparavant. L'itération s'effectue sur la longueur de la liste $\mathcal{L}$. En effet, tant que la liste possède des éléments, le processus de construction de la matrice s'effectue. Nous remarquons que la liste $\mathcal{L}$ atteindra la liste vide en un temps fini puisqu'à chaque étape de la construction de la matrice $\mathscr{M}$, un élément de la matrice est supprimé, ce qui réduit la taille de la matrice $\mathscr{M}$ de un. Même si, à chaque étape de la construction de $\mathscr{M}$, un ou plusieurs éléments peuvent être ajoutés, l'ajout n'est pas infini. En effet, le nombre de services web est fini, tout comme leurs nombres de pré-conditions et effets, ce qui implique que la découverte de nouveaux services aura une fin, et ainsi la réduction de la liste $\mathcal{L}$ sera assurée par la suppression d'un

de ses éléments à chaque étape. L'arrêt de l'algorithme 9 est donc assuré par l'arrêt des trois étapes traitées ci-dessus.

2. **Preuve de correction:**

   La correction de l'algorithme 9 est vérifiée si et seulement si chaque élément de la matrice $\mathscr{M}$ est instancié par un ensemble de couple $(S,\ score)$ où S représente un ensemble de services web alors que $score$ représente le score pour un élément donné de la matrice. Par l'algorithme 6, la matrice $\mathscr{M}$ est instanciée puis initialisée. Cependant montrons que chaque élément de la matrice $\mathscr{M}$ est modifié par l'algorithme 9 via un modification correcte. Un élément de la matrice $\mathscr{M}$ est modifié par l'algorithme 9 si et seulement s'il vérifie un des deux cas suivant:

   - $e \in \mathcal{KB}$;
   - $e \in \bigcup_{i=1}^{n} pre(Ws_i) \cup \beta_L$.

   Le premier cas reflète la présence de e dans la base de connaissance $\mathcal{KB}$, ainsi une instance de e est connue. Ainsi:

   $$\forall X \in \{1, ..., p\},\ \mathscr{M}(X,\ e).score = 1\ et\ \mathscr{M}(X,\ e).set = \mathcal{KB}$$

   Le deuxième cas reflète la présence de e dans l'ensemble des buts locaux β_L ou $\bigcup_{i=1}^{n} pre(Ws_i)$, ce qui signifie qu'une instance de e est à trouver. Trouver une instance de e revient donc à trouver un service web vérifiant les post-conditions e. Ainsi dans le but de trouver e, nous recherchons les services web ayant une similitude Sim_T strictement positive avec un service ayant les effets e. Il est donc correct de rechercher les services web au sens de la similitude Sim_T définie auparavant. Une fois ces services retrouvés, le processus itère sur ceux-ci, puis sur les lignes et colonnes de la matrice de Matching $\mathscr{M}$, afin de modifier les éléments de la matrice $\mathscr{M}$ susceptibles de satisfaire une similarité Sim_T strictement positive. Le troisième cas reflète l'absence de e dans l'ensemble des buts locaux β_L. Cette absence implique que e appartient à une pré-condition d'un service web. Il est donc pertinent de ne pas calculer la similarité classique entre un service ayant un effet e et un service web mais de calculer la similarité suivante:

   $$Sim_T(S_{AIC}^{pre}(WS(pre\ ,\ e)), WS_k)$$

   Cette similarité permet de calculer la similarité entre un service web ayant des effets pre et des pré-conditions à déterminer avec un service proche WS_k. Ce dernier cas est nécessaire pour la construction de la matrice de Matching $\mathscr{M}$ puisqu'il permet de prendre en compte les services inverses au sens des pré-conditions. Afin d'achever la preuve de la correction de l'algorithme 9, nous introduisons le cas de non-determinisme. En effet plusieurs services web peuvent apparaître pour un élément de la matrice $\mathscr{M}$, il est donc nécessaire de prendre en compte la multiplicité des services web à l'aide du test suivant:

   $$\mathscr{M}(p_c,\ e) \neq (\emptyset,\ 0)$$

   En effet un tel test permet de vérifier l'existence d'un service web pour un élément donné de la matrice $\mathscr{M}$. Si ce test est vérifié, nous procédons par un simple ajout afin de mettre à jour la matrice $\mathscr{M}$. Une fois la liste $\mathcal{L}$ vide, nous retournons la matrice $\mathscr{M}$ considérée comme finale et correcte.

3. **Analyse de la complexité:**

   La complexité de l'algorithme 9 est fonction des trois étapes introduites lors de la preuve de l'arrêt. L'étape *ligne 10* est fonction du nombre de lignes de la matrice $\mathscr{M}$. Ainsi si $e \in \mathcal{KB}$,

la complexité est évaluée à $\theta(p)$. Si nous reprenons le formalisme introduit dans l'analyse de complexité de l'algorithme 8 alors la complexité de l'étape *ligne 16* est de l'ordre de $\theta(np))$ dans le pire des cas. L'étape *ligne 7* consiste à dérouler tous les éléments de la liste $\mathcal{L}$ soit dans la pire des cas tous les effets, pré-conditions des services web ainsi que les buts locaux, d'où $\theta(q))$. Ainsi la complexité finale de l'algorithme 9 est de l'ordre de $\theta(npq))$ soit dans le pire des cas cubique. Cependant dans le cas général, la taille de la liste $\mathcal{L}$ ne tend pas vers q. Ainsi cet algorithme possède un complexité moins importante que celle de l'algorithme 8. Nous pouvons conclure que la construction de la matrice $\mathscr{M}$ à l'aide de l'algorithme 9 est plus efficace.

F.4 Construction de $\mathscr{M}$ sur un exemple simple

Dans cette section, nous présenterons un exemple de construction de la matrice $\mathscr{M}$ portant sur le domaine du cinéma.

Tout d'abord supposons un ensemble de services web $S_{WebServices}$ tel que:

$$S_{WebServices} = \{s_1,\ s_2,\ s_3,\ s_4,\ s_5,\ s_6,\ s_7\ \}$$

où les sept différents services web sont explicités par leurs entrées et sorties[5] respectives. Ainsi nous définissons les sept services web comme suit:

- s_1(in Movie, in UserProfile, out Cinema);
- s_2(in GeographicArea, in MovieGenre, out FamilyCinema);
- s_3(in FrenchMovie, in GeographicArea, out FrenchCinema);
- s_4(in PhoneNumber, out Location);
- s_5(in Director, out Black&WhiteMovie);
- s_6(in Director, out ColorMovie);
- s_7(in Year, out LocalMovie).

Nous considérons un unique but local noté $b_l \in \beta_L$. Le but sera d'obtenir un Cinéma (son adresse, nom...). De plus nous considérons la base de connaissance KB définie dans le tableau F.1.

Concept	Instance
Director	Tom Paul
UserProfile	my_profile
MovieGenre	Comics
Year	2005
PhoneNumber	+33677777777

Table F.1: Base de connaissance $\mathcal{KB}$.

[5] Afin de simplifier au maximum la construction de la matrice de Matching $\mathscr{M}$, nous confondrons Pré-conditions (resp. Post-conditions) et Input (resp. Output).

Afin de nous borner à la construction de la matrice de Matching $\mathscr{M}$, nous n'expliciterons pas les détails du calcul de similitude. Cependant la connaissance de la table de similitude est essentielle pour la construction de la matrice $\mathcal{KB}$. Ainsi nous présentons les résultats de la similitude des services web. Le tableau F.2 explicite[6] les résultats nécessaires au cas $e \in \beta_L$ de l'algorithme 9. Le tableau F.2 indique les services capables de résoudre le but *"Cinema"*.

Sim_T	s_1	s_2	s_3
Ws($\mathcal{KB}$, Cinema)	0.9	0.7	0.6

Table F.2: Résultats nécessaires pour le cas $e \in \beta_L$ de l'algorithme 9.

Le tableau F.3 explicite les résultats nécessaires au cas $e \notin \beta_L$ de l'algorithme 9.

Sim_T	s_4	s_5	s_6	s_7
$Ws_X(\mathcal{KB},\ Movie)$		0.6	0.5	0.3
$Ws_Y(\mathcal{KB},\ GA)$	0.8			
$Ws_T(\mathcal{KB},\ FrenchMovie)$			0.5	

Table F.3: Résultats nécessaires pour le cas $e \notin \beta_L$ de l'algorithme 9.

Les résultats des tableaux F.2 et F.3 ont été obtenus à l'aide des ontologies du domaine, de la description des concepts, et de la fonction de similitude sim_T. L'élément "$\mathcal{KB}$" dans le tableau F.3 signifie que nous posons des contraintes sur les services à retrouvé (au niveau des pré-conditions de ceux-ci). Cependant il est fort possible d'imposer aux services à retrouver de vérifier des propriétés plus complexes afin d'élaguer l'espace de recherche et donc d'affiner la recherche.

Nous avons donc défini le contexte dans lequel nous travaillons. Avant de construire définitivement la matrice de Matching $\mathscr{M}$, nous devons utiliser l'algorithme 6 qui est un pré requis de l'algorithme 9 afin de créer et initialiser les outils nécessaires à la construction de $\mathscr{M}$. L'algorithme 6 produit donc la liste:

$$\mathcal{L} = \{Cinema\}$$

Avant de construire la matrice $\mathscr{M}$ initialisée à $M_\emptyset$, déterminons p et q référençant respectivement le nombre de lignes et de colonnes de $\mathscr{M}$. p est déterminé à l'aide de la formule (F.1). Nous obtenons ainsi $p = 8$. q est déterminé à l'aide de la formule (F.2). Nous obtenons ainsi $q = 9$. Ainsi $\mathscr{M} \in M_{8,9}(S_{WebServices} * [0,\ 1])$. $\mathscr{M}$ est donc représenté par une matrice nulle noté $M_\emptyset$ avec huit lignes et neuf colonnes où les lignes et colonnes ont les correspondances suivantes:

- La pré-condition *"Director"* identifie indépendamment la ligne 1 et la colonne 1;
- La pré-condition *"Movie"* identifie indépendamment la ligne 2 et la colonne 2;
- La pré-condition *"UserProfile"* identifie indépendamment la ligne 3 et la colonne 3;
- La pré-condition *"GeographicArea"* identifie indépendamment la ligne 4 et la colonne 4;
- La pré-condition *"MovieGenre"* identifie indépendamment la ligne 5 et la colonne 5;
- La pré-condition *"PhoneNumber"* identifie indépendamment la ligne 6 et la colonne 6;

[6] Les résultats du tableau F.2 et F.3 sont calculés à l'aide de la fonction de calcul de similitude sim_T.

- La pré-condition *"FrenchMovie"* identifie indépendamment la ligne 7 et la colonne 7;
- La pré-condition *"Year"* identifie indépendamment la ligne 8 et la colonne 8;
- Le but local b_l *"Cinema"* identifie la colonne 9.

Après avoir introduit les éléments nécessaires à la construction de la matrice de Matching $\mathscr{M}$, nous pouvons élaborer la construction effective de $\mathscr{M}$ à l'aide de l'algorithme 6 et des outils présentés jusqu'ici. Ainsi déroulons l'algorithme 9 afin de construire la matrice $\mathscr{M}$ finale.

F.4.1 Étape 1: $\mathcal{L} = \{Cinema\}$

D'après l'algorithme 9 $e \leftarrow Cinema$, cependant e est assimilé à un but local puisque $e \in \beta_L$. Ainsi nous considérons le cas *ligne 10*. Ensuite, nous recherchons tous les services web Ws_k tels que:

$$Sim_T(WS(\mathcal{KB}\ ,\ e), Ws_k) \geq 0$$

Ces services sont retrouvés à l'aide du tableau F.2 Ainsi $WS(\mathcal{KB}\ ,\ e)$ possède une similitude positive avec s_1, s_2, et s_3. Donc pour chacune des pré-conditions p_c de ces trois services, nous remplissons la matrice $\mathscr{M}$ si et seulement si $p_c \notin \mathcal{L}_{temp}$. Ainsi grâce au service web s_1 retrouvé, nous obtenons:

- $\mathscr{M}$(Movie, Cinema) $\leftarrow (s_1,\ 0.9)$;
- $\mathscr{M}$(UserProfile, Cinema) $\leftarrow (s_1,\ 0.9)$.

Nous itèrons ce processus pour trouver $\mathscr{M}$(GeographicArea, Cinema), $\mathscr{M}$(MovieGenre, Cinema), $\mathscr{M}$(FrenchMovie, Cinema).
Ainsi à la fin de cette étape, les listes sont respectivement:

$$\mathcal{L} = \{Movie,\ UserProfile,\ GeographicArea,\ MovieGenre,\ FrenchMovie\}$$

$$\mathcal{L}_{temp} = \{Cinema,\ Movie,\ UserProfile,\ GeographicArea,\ MovieGenre,\ FrenchMovie\}$$

F.4.2 Étape 2

Si nous considérons la nouvelle liste $\mathcal{L}$, alors e est assimilé à l'élément *"Movie"* qui ne fait pas partie de la base de connaissance $\mathcal{KB}$, ni de l'ensemble des buts locaux β_L. Ainsi nous nous trouvons dans le cas où $e \notin \beta_L$ et $e \notin \mathcal{KB}$. Nous recherchons donc tous les services web Ws_k tels que:

$$Sim_T(S^e_{AIC}(WS(e,\ post)), WS_k) \geq 0 \qquad i.e \qquad Sim_T((WS(\mathcal{KB},\ e)), WS_k) \geq 0$$

Grâce au tableau F.3, nous retrouvons les services s_5, s_6, et s_7 ayant une similitude Sim_T positive avec $(WS(pre,\ Movie)$. Donc pour chacune des pré-conditions p_c de ces trois services, nous remplissons la matrice $\mathscr{M}$ si et seulement si $p_c \notin \mathcal{L}_{temp}$. Ainsi grâce au service web s_5 retrouvé, nous obtenons:

- $\mathscr{M}$(Director, Movie) $\leftarrow (s_5,\ 0.6)$.

Nous itérons ce processus pour trouver $\mathscr{M}$(Year, Movie).
Ainsi à la fin de cette étape, les listes sont respectivement:

$$\mathcal{L} = \{UserProfile,\ GeographicArea,\ MovieGenre,\ FrenchMovie,\ Director,\ Year\}$$

F.4.3 Étape 3

Si nous considérons la nouvelle liste $\mathcal{L}$, alors e est assimilé à l'élément *"UserProfile"* qui ne fait pas partie de la base de connaissance $\mathcal{KB}$. Ainsi d'après l'algorithme 9:

$$\mathscr{M}(l,\ e) = (\mathcal{KB},\ 1) \qquad \forall l \in \{1, ..., p\}$$

F.4.4 Résultat final

	Dir	Movie	UP	GA	MG	PN	FM	Year	Cinema
Dir	$(\mathcal{KB}, 1)$	$\{(s_5, 0.6)$ $(s_6, 0.5)\}$	$(\mathcal{KB}, 1)$	$(\emptyset, 0)$	$(\mathcal{KB}, 1)$	$(\mathcal{KB}, 1)$	$(\emptyset, 0)$	$(\mathcal{KB}, 1)$	$(\emptyset, 0)$
Movie	$(\mathcal{KB}, 1)$	$(\emptyset, 0)$	$(\mathcal{KB}, 1)$	$(\emptyset, 0)$	$(\mathcal{KB}, 1)$	$(\mathcal{KB}, 1)$	$(\emptyset, 0)$	$(\mathcal{KB}, 1)$	$(s_1, 0.9)$
UP	$(\mathcal{KB}, 1)$	$(\emptyset, 0)$	$(\mathcal{KB}, 1)$	$(\emptyset, 0)$	$(\mathcal{KB}, 1)$	$(\mathcal{KB}, 1)$	$(\emptyset, 0)$	$(\mathcal{KB}, 1)$	$(s_1, 0.9)$
GA	$(\mathcal{KB}, 1)$	$(\emptyset, 0)$	$(\mathcal{KB}, 1)$	$(\emptyset, 0)$	$(\mathcal{KB}, 1)$	$(\mathcal{KB}, 1)$	$(\emptyset, 0)$	$(\mathcal{KB}, 1)$	$\{(s_2, 0.7)$ $(s_3, 0.6)\}$
MG	$(\mathcal{KB}, 1)$	$(\emptyset, 0)$	$(\mathcal{KB}, 1)$	$(\emptyset, 0)$	$(\mathcal{KB}, 1)$	$(\mathcal{KB}, 1)$	$(\emptyset, 0)$	$(\mathcal{KB}, 1)$	$(s_2, 0.7)$
PN	$(\mathcal{KB}, 1)$	$(\emptyset, 0)$	$(\mathcal{KB}, 1)$	$(s_4, 0.8)$	$(\mathcal{KB}, 1)$	$(\mathcal{KB}, 1)$	$(\emptyset, 0)$	$(\mathcal{KB}, 1)$	$(\emptyset, 0)$
FM	$(\mathcal{KB}, 1)$	$(\emptyset, 0)$	$(\mathcal{KB}, 1)$	$(\emptyset, 0)$	$(\mathcal{KB}, 1)$	$(\mathcal{KB}, 1)$	$(\emptyset, 0)$	$(\mathcal{KB}, 1)$	$(s_3, 0.6)$
Year	$(\mathcal{KB}, 1)$	$(s_7, 0.3)$	$(\mathcal{KB}, 1)$	$(\emptyset, 0)$	$(\mathcal{KB}, 1)$	$(\mathcal{KB}, 1)$	$(s_7, 0.5)$	$(\mathcal{KB}, 1)$	$(\emptyset, 0)$

Table F.4: Matrice $\mathscr{M}$ finale.

Nous avons traité tous les cas possibles de l'algorithme, à savoir "$e \in \mathcal{KB}$", "$e \in \beta_L$", et "$e \notin \mathcal{KB}$ *et* $e \notin \beta_L$". Nous ne traiterons pas toute la liste $\mathcal{L}$ pour des raisons de place cependant nous décrivons la matrice finale $\mathscr{M}$ en figure F.4.

F.5 Synthèse

Dans ce chapitre nous avons introduit la matrice de liens sémantiques permettant de résoudre le problème de composition de services. La construction d'une telle matrice nécessite la connaissance d'une fonction de similitude Sim_T introduite lors du chapitre précédant. Le chapitre suivant aura pour but de résoudre le problème de composition de services au moyen de la matrice de liens sémantiques. Afin de résoudre un tel problème, nous comparons notre problème à un problème de planification.

Appendix G

Construction du plan final

G.1 Introduction

Après avoir introduit le processus de construction de la matrice de matching (ou de liens sémantiques) $\mathscr{M}$ aidant à la composition automatisée de services web, nous proposons d'étudier l'algorithme permettant la composition finale des services web retrouvés. L'algorithme proposé s'appuie sur le domaine de la planification en IA. Ainsi nous proposons de résoudre un problème de planification pour résoudre notre problème de composition. Tout d'abord nous présenterons les définitions importantes relatives au domaine de la planification en IA, ensuite nous analyserons notre problème de planification. Enfin nous présenterons notre algorithme de résolution de composition de services.

G.2 La planification dans le cadre de l'IA

G.2.1 La planification en IA

La planification portant sur le domaine de l'intelligence artificielle a débuté dans le courant de l'année 1963 lors de la présentation de *GPS* (General Problem Solver) par [146]. En général, un planificateur IA produit un ensemble d'actions ordonnées ayant pour but la modification de l'état du monde afin d'obtenir l'état du monde désiré.

Le problème de planification est vu comme un problème de construction automatique ou semi-automatique d'ordre sur les actions disposées de sorte que l'exécution de ces actions a pour but de modifier l'état initial du monde en un autre état appelé état final du monde dans lequel certains buts ont été réalisés. Cet ordre est typiquement assimilé à un "ordre partiel". En effet, l'ordre retrouvé peut être assimilé à un ordre partiel où des séquences d'actions sont effectuées en parallèle. Ainsi ces séquences peuvent être effectuées dans n'importe quel ordre tout en permettant la réalisation des buts désirés. Le planificateur ou générateur de plans est le système qui le produit.

Problème de planification

Le processus de planification nécessite:

- une description du problème à résoudre:
  - une description de l'état initial du monde en question (l'état initial),

  - une description du but à atteindre (le but),
  - une description des opérations qu'il est possible de réaliser dans le monde (parfois de différentes granularités : actions, opérateurs, procédures, tâches...);
- une procédure formalisée qui manipule ces descriptions afin de produire un plan solution;
- si possible, des connaissances permettant de guider la recherche d'un plan solution (connaissances indépendantes ou liées au domaine ou au problème posé).

Propriétés des plans

Un plan est *correct* si et seulement si son exécution permet, partant de l'état initial, d'atteindre le but.

Un plan est *complet* si et seulement si chaque pré-condition d'une étape du plan est satisfaite par un effet d'une étape antérieure du plan (aucune pré-condition n'est dite "ouverte"). Plus formellement, une étape du plan S_i satisfait les pré-conditions p d'une étape du plan S_j si $S_i \prec S_j$ avec $p \in \texttt{EFFECTS}(S_i)$.

Un plan est *consistant* si et seulement s'il n'y a pas de contradictions lors de l'ordonnancement[1] des contraintes. Il y a contradiction si $S_i \prec S_j$ et $S_j \prec S_i$.

Un plan est une *solution* au problème de planification si et seulement si celui-ci est correct, complet et consistant.

Propriétés des planificateurs

Un planificateur est *sain* si et seulement si, étant donné un problème de planification, et étant donné le formalisme de description de problèmes, tous les plans qu'il est susceptible de produire pour résoudre ce problème sont des plans solutions.

Un planificateur est *complet* si et seulement si, étant donné un problème de planification, et étant donné le formalisme de description de problèmes, il produit un plan solution lorsqu'il existe.

Un planificateur est *optimal* (suivant un critère particulier qui mesure la qualité des plans) si et seulement si les plans solutions qu'il produit sont optimaux selon ce critère (nombre d'actions, nombre de niveaux, degré de parallélisme, temps d'exécution, quantité de ressources consommées, qualité de Matching...).

G.2.2 Les modèles élaborés par l'IA

Calcul de situation (Planification déductive)

En calcul de situation[133], une situation représente l'état du monde à l'instant t (l'état du monde est exprimé en termes de fonctions et de relations relatives à une situation particulière) et une action représente une opération permettant de changer de situations. Dans le calcul de situations, nous nous intéressons à l'état du monde après avoir accompli des actions.

1. Description des actions

   Chaque action a est représentée par deux ensembles distincts:

   - l'ensemble des possibilités où chaque élément est de la forme $preconditions \Rightarrow Possible(a,\ s)$ signifiant que l'action a est possible dans la situation s;

[1] $\prec$ est une relation transitive permettant d'ordonnancer le plan.

- l'ensemble des effets où chaque élément est de la forme $Possible(a,\ s) \Rightarrow effet$ signifiant que l'effet décrit les changements résultant de l'action a.

Synthèse de plans: Système STRIPS

STRIPS[64][2] a été adopté par la communauté de recherche comme la base des formalismes de planification utilisée aujourd'hui. Différentes extensions [125, 136] de cette planification existe mais la description de haut niveau présentée dans *STRIPS* est commune à toutes ces approches.

Dans *STRIPS* le monde est décrit en terme de son état. Ainsi le monde est décrit au moyen de trois listes de variables: une liste de pré-conditions, une liste d'éléments à effacer, et une liste d'éléments à ajouter. La liste des pré-conditions permet de vérifier ce qui doit être vérifié avant toute action. Les listes des éléments à effacer et à ajouter renseignent sur les modifications de l'état du monde.

1. Représentation des états

   Un état E du monde de la planification est représenté par un ensemble fini de formules atomiques (conjonction d'atomes clos) sans symbole de variables et de fonctions. Tout état ne contient pas de propositions négatives. *STRIPS* satisfait l'hypothèse du monde fermé: *"toute proposition qui n'est pas exprimée dans un état est fausse dans cet état"*. Une formule atomique de base est aussi appelée un fluent.

2. Représentation du but

   Il est représenté par un état partiellement spécifié. Un état satisfait le but s'il contient tous les atomes du but. (ex: $Cinema \cap cheap$ est un but, $Cinema \cap cheap \cap commercial$ est un état satisfaisant le but). Tout but ne contient pas de propositions négatives.

3. Représentation des opérateurs

   Un opérateur o est un modèle d'action. Il est représenté par son nom et un triplet $< pr,\ ad,\ de >$ où pr, ad et de sont des ensembles finis de formules atomiques. $Prec(o)$, $Add(o)$, $Del(o)$ dénotent respectivement les ensembles pr, ad, de de l'opérateur o. Une action, dénotée par a, est une instance de base d'un opérateur o (toutes les variables de o sont instanciées).

4. Représentation des actions

   Chaque action (ou schéma d'action) est spécifiée par un nom, un ensemble de pré-conditions et un ensemble d'effets. Les pré-conditions doivent être satisfaites avant l'action et les effets sont impliqués après l'action.

Action(aller(a, x, y),
`PRECOND`: *lieu(a, x)* $\wedge$ *acote(x; y)*
`EFFECT`: $\neg$*lieu(a, x)* $\vee$ *lieu(a, y))*

Figure G.1: Exemple d'action avec le formalisme *STRIPS*.

[2] STandford Research Institute Problem Solver.

5. Résultat d'une action

   L'application de l'action a sur un état s donne un état s' qui est le même que s sauf qu'il a été agrémenté des litéraux positifs des effets dans s' et réduit des litéraux négatifs des effets de s'.

6. Problème de planification

   Un problème de planification est un triplet $< O,\ I,\ B >$ où :

   - O dénote un ensemble fini d'opérateurs utilisables dans le domaine de la planification considéré,
   - I est l'état initial du problème, il est représenté par un ensemble fini de fluents,
   - B est le but du problème, il est représenté par un ensemble fini de fluents.

7. Solution au problème de planification

   Une solution au problème de planification est un plan solution.

Décomposition en tâches hiérarchisées: Système NOAH

La planification hiérarchique HTN[181][3] est une méthodologie de planification (type Intelligence Artificielle) qui construit un plan par décomposition de tâches. C'est un processus dans lequel le système de planification décompose des tâches en plus petites sous tâches jusqu'à obtention de tâches primitives (ou services primitifs). Ainsi ces dernières pourront être exécutées directement. Ce découpage en sous tâches permet d'obtenir la connaissance de l'état du monde à chaque étape. La planification hiérarchique est caractérisée par le fait qu'elle construit le plan d'actions à des niveaux successifs de détail ou d'abstraction. La motivation principale, qui a conduit à cette approche, est la réduction de l'espace de recherche à chaque étape d'affinement du plan, donc un gain d'efficacité.

G.2.3 Méthode de planification

Dans la suite du chapitre nous utiliserons le formalisme introduit par *STRIPS*.

Planification linéaire (ordre total)

Le planificateur le plus simple est le planificateur linéaire: il procède par exploration de l'espace des états à l'aide d'un algorithme de recherche.

Chaînage avant (data-driven)

La planification "en avant" explore un espace d'états. Le planificateur effectuant une telle planification est appelé "progressif ". Le "Chaînage avant" a pour but de considérer des séquences d'actions jusqu'à ce que nous trouvions une séquence qui mène à l'état de solution. Cependant l'application d'une telle méthode est difficile car le nombre d'actions possibles est souvent grand (en général, de nombreuses actions sont applicables à un état donné ce qui implique un facteur de branchement élevé).

[3] hierarchical planning formalism: HTN

Méthodes de recherche	Espace de recherche
Planification “avant”	États
Planification “arrière”	Objectifs
Planification non linéaire	Plans

Table G.1: Modèle de planification et espaces de recherche.

Chaînage arrière (goal-driven ou regression du but)

La planification en chaînage arrière explore un espace d'objectifs à partir de l'objectif original vers un but correspondant à l'état initial. Le planificateur effectuant une telle planification est appelé “régressif”.

STRIPS effectue une planification linéaire par “chaînage arrière”, pour cela elle utilise une structure de pile pour organiser les sous problèmes et sélectionner lequel poursuivre.

Chaînage avant vs Chaînage arrière

En général, il y a beaucoup moins d'actions pertinentes pour un but donné qu'il n'y a d'actions applicables. Ainsi le facteur de branchement pour le chaînage arrière est moindre par rapport à la planification ”en avant”. Parfois nous ne pouvons pas connaître toutes les variables pour complètement instancier un opérateur en chaînage arrière. Néanmoins les longueurs des solutions (en nombre d'actions) sont identiques.

Un plan séquentiel P est une séquence finie (éventuellement vide) d'actions notée $< a_1, a_2, ..., a_n >$ (notée $<>$ si P est la séquence vide).

Planification non linéaire (ordre partiel)

Le vrai problème posé par l'interférence entre buts ne se trouve pas au niveau des actions elles-mêmes, mais de leur ordre. La planification non linéaire consiste à trouver d'abord les sous plans pour chaque but, et ensuite trouver l'ordre qui convient. La planification non linéaire explore un espace de plans. Dans la planification non linéaire, les plans représentent des ordres partiels. L'idée générale est le “least committment”, c'est-à-dire ne pas faire de choix avant que ce soit vraiment nécessaire.

G.2.4 Synthèse

La planification est un problème très difficile, car il n'y a pas de limite supérieure à la complexité du plan nécessaire pour réaliser un but donné.

G.3 Composition de services et planification en IA

G.3.1 Formalisme du problème de composition automatique de services

Le problème de composition automatique de services web sera vu comme un problème de planification $< S_{WebServices}, \mathcal{KB}, \beta_G >$ (voir figure G.2) où :

- $S_{WebServices}$ dénote un ensemble fini de services web utilisables dans le domaine de la planification considéré;

- $\mathcal{KB}$ est la base de connaissance, il est représenté par un ensemble fini de concepts et d'instances;
- β_G est le but du problème, il est représenté par un ensemble fini de concepts.

β_G étant décomposable en sous buts locaux β_{L_i}, le problème de composition automatique de services web se ramène à un problème de planification non linéaire de $< S_{WebServices}, \mathcal{KB}, \beta_G >$ où $\beta_G = \{\beta_{L_i} \mid \beta_{L_i} \in \beta_L\}$. Ainsi résoudre le probème de planification $< S_{WebServices}, \mathcal{KB}, \beta_G >$ revient à résoudre le problème de planification non linéaire de $< S_{WebServices}, \mathcal{KB}, \beta_{L_i} >$ avec $\beta_{L_i} \in \beta_L$. En effet la planification par réduction de problème semble appropriée à notre problème de composition. Il est donc plus facile de résoudre un problème complexe en le réduisant en un ensemble de sous problèmes plus faciles à résoudre.

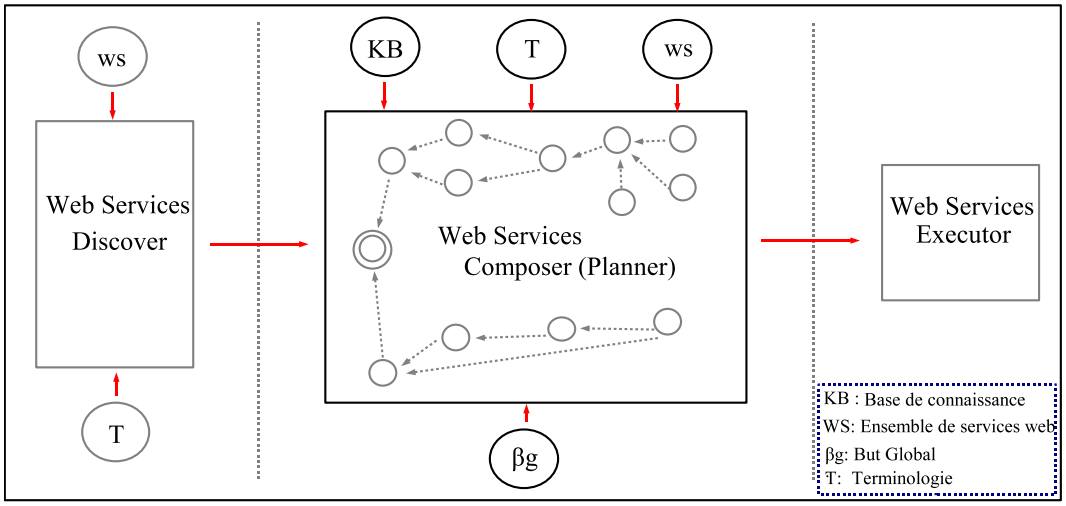


Figure G.2: Problème de planification.

Définition 12. *(Plan vide) Un plan vide noté <> est l'unique plan ne contenant aucune étape et ne résolvant aucun but.*

Propriété 11. *Le problème de planification $< S_{WebServices}, \mathcal{KB}, \emptyset >$ possède un unique plan solution optimal <>.*

Proof. Montrons que <> est un plan correct, complet, consistant et optimal du problème $< S_{WebServices}, \mathcal{KB}, \emptyset >$.

Notons qu'à partir de l'état initial représenté par la base de connaissance, il est possible d'atteindre le but $\emptyset$, puisque ce but est toujours satisfait quelque soit le plan envisagé. Ainsi nous avons montré que partant de l'état initial le plan <> permet d'atteindre le but final $\emptyset$. Le plan <> est donc un plan correct pour le problème de planification $< S_{WebServices}, \mathcal{KB}, \emptyset >$.

Le plan <> est complet pour le problème de planification $< S_{WebServices}, \mathcal{KB}, \emptyset >$, puisque celui-ci n'est constitué d'aucune étape. En effet le plan n'est constitué d'aucune pré-condition "ouverte".

Le plan <> ne contenant aucune étape, aucune contradiction lors de l'ordonnancement des contraintes ne peut être référencée. Ainsi le plan <> est consistant.

Montrons que le plan <> est l'unique plan résolvant le problème de planification $< S_{WebServices}, \mathcal{KB}, \emptyset >$. D'après la définition du "Plan vide", <> est l'unique plan ne contenant aucune étape et ne résolvant aucun but. Ainsi par analogie, <> est l'unique plan résolvant le problème de planification $< S_{WebServices}, \mathcal{KB}, \emptyset >$.

Le plan solution du problème de planification $< S_{WebServices}, \mathcal{KB}, \emptyset >$ étant unique, il est optimal. □

G.3.2 Solution envisagée

Ayant connaissance de l'ensemble des buts locaux β_L, notre planificateur procèdera par "chaînage arrière", procédant par l'exploration de l'espace des objectifs, par régression jusqu'à découverte des états initiaux (i.e. les pré-conditions du plan sont satisfaites soit par la base de connaissance $\mathcal{KB}$, soit par un service web (une action)). De plus, le plan retourné devra être solution, et optimal au sens de la métrique Sim_T introduite lors du chapitre E.

Définition 13. *(Plan complet)*
Un plan est complet *si et seulement si chaque pré-condition d'une étape du plan est satisfaite par un effet d'une étape antérieure du plan ou si elle est satisfaite par la base de connaissance $\mathcal{KB}$ (aucune pré-condition n'est dite "ouverte"). Plus formellement, une étape du plan S_i satisfait les pré-conditions p d'une étape du plan S_j si $S_i \prec S_j$ avec $p \in$ `EFFECTS`(S_i) ou si $p \in \mathcal{KB}$.*

Nous avons donc introduit la notion de plan complet, définition sensiblement différente de celle introduite par la communauté de planification en IA. Cependant les définitions de correction et consistance d'un plan sont rigoureusement identiques à celles introduites par la communauté de planification en IA. De même un plan est une *solution* au problème de planification si et seulement si celui-ci est correct, complet et consistant.

Définition 14. *(Service web post-pertinent[4])*
Une service web est post-pertinent pour réaliser un objectif donné si une des propositions de ses effets correspond à une proposition d'un sous objectif.

D'après la définition précédente les services web entrant en ligne de compte dans le plan solution sont ceux qui ont le potentiel de réaliser un but ou sous but. Leurs pré-conditions définiront les sous buts du prochain problème à résoudre.

Définition 15. *(Service web pre-pertinent)*
Une service web est pre-pertinent pour réaliser un objectif donné si toutes les propositions de ses pré-conditions sont satisfaites par un ensemble fini d'éléments de $\mathcal{KB}$.

Ainsi d'après la définition précédente, un plan complet est constitué de services web pre-pertinents permettant de résoudre un but ou sous but du plan.

Définition 16. *(Service web pertinent)*
Une service web est pertinent si et seulement si celui-ci est pre-pertinent et post-pertinent.

Propriété 12. *Un plan est complet si et seulement il est constitué de services web pertinents, et/ou pre-pertinents, et post-pertinents.*

Proof. Traitons les deux implications de la propriété.

[4] Définition proche d'*"action pertinente"* introduite par la communauté de planification en IA

→ D'après la définition 13, un plan est complet si chaque pré-condition d'une étape du plan est satisfaite par un effet d'une étape antérieure du plan ou si elle est satisfaite par la base de connaissance $\mathcal{KB}$. Supposons deux cas:

1) Supposons qu'une pré-condition d'une étape du plan est satisfaite par un effet d'une étape antérieure du plan alors d'après la définition 14, il existe un service web post-pertinent satisfaisant la pré-condition de cette étape.
2) Supposons qu'une pré-condition d'une étape du plan est satisfaite par la base de connaissance $\mathcal{KB}$ alors d'après la définition 15, il existe un service web pre-pertinent satisfaisant la pré-condition de cette étape.

← Supposons deux cas:

1) Supposons que le plan proposé ne soit constitué que de services web pertinents alors d'après les définitions 14, 15 et 16, tous ces services ont des pré-conditions satisfaites par $\mathcal{KB}$ et ont au moins un effet correspondant à une proposition d'un sous objectif. Ainsi chaque sous objectif est résolu par exactement un service web pertinent du fait de sa définition. Ainsi chaque pré-condition d'une étape du plan est satisfaite par la base de connaissance $\mathcal{KB}$ d'après la définition 15. Ainsi le plan est complet.
2) Supposons que le plan proposé ne soit constitué que de services web pre-pertinents et post-pertinents alors d'après la définition 14, et 15 il existe des services web dont les effets satisfont les pré-conditions d'un sous objectif et il existe des services web dont les pré-conditions sont satisfaites par $\mathcal{KB}$. Cependant cette dernière affirmation valide la définition de plan complet.

□

Recherche des services web pertinents, pre-pertinents, et post-pertinents

Afin de trouver un plan solution à notre problème de planification, il est essentiel de retrouver les services web pertinents, pre-pertinents, et post-pertinents permettant d'assurer la complétude du plan. Afin de retrouver ces services web, nous proposons un algorithme procédant par "chaînage arrière". Ainsi, à partir des objectifs à atteindre nous construirons le plan complet à l'aide de la matrice de Matching $\mathscr{M}$ renseignant sur les liens sémantiques à mettre en oeuvre. Nous rappelons que les liens sémantiques sont pondérés, et que chaque valeur non nulle d'un élément $\mathscr{M}(i,\ j)$ de la matrice de Matching $\mathscr{M}$ signifie l'existence d'un service web Ws_x dont les effets sont plus spécifiques (ou équivalents: dépendant de la fonction de Matching appliquée) que les pré-conditions incarnées par j. Autrement dit il existe un service web Ws_x tel que

$$Ws_x \xrightarrow{j} Ws(j,\ ?)$$

signifiant qu'il existe un lien sémantique entre Ws_x et $Ws(j,\ ?)$.

Définition 17. *(Liens sémantiques) Les liens sémantiques mettent en évidence la manière dont le plan satisfait les buts à partir des conditions initiales (figure G.3).*

Un lien sémantique s'écrit comme $S_i \xrightarrow{c} S_j$ et se lit *"les post-conditions de S_i satisfont les pré-conditions de S_j"*. Les liens sémantiques sont utiles pour mémoriser les buts de chaque étape du plan. Dans ce cas précis, le but de S_i est de satisfaire les pré-conditions c de S_j.

D'après la définition précédente, nous pouvons affirmer que la matrice de Matching $\mathscr{M}$ peut être assimilée à la matrice des liens sémantiques.

Propriété 13. *Les services web pertinents, pre-pertinents, et post-pertinents sont connus au moyen de la matrice de Matching $\mathscr{M}$ (ou matrice des liens sémantiques).*

Proof. Trivial d'après la construction de la matrice de Matching. $\mathcal{M}$. □

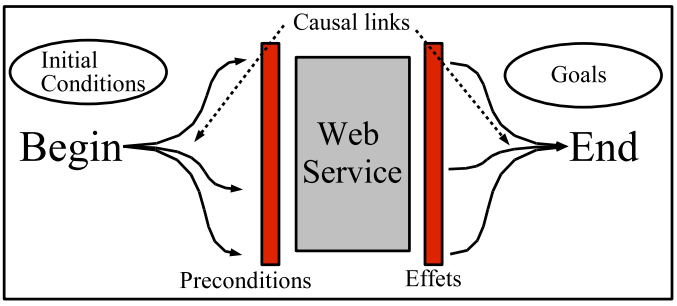


Figure G.3: Liens sémantiques.

G.4 Construction du plan final

Étant donnée la matrice des liens sémantiques $\mathscr{M}$, nous présentons l'algorithme permettant de construire la composition optimale de services web au sens de la fonction de similitude Sim_T introduite lors du chapitre E. Nous proposons un algorithme procédant par "chaînage arrière" pour résoudre le problème de planification $< S_{WebServices}, \mathcal{KB}, \beta_G >$. Le planificateur proposé devra posséder les propriétés de complétude, consistance, correction et d'optimalité (au sens de la fonction Sim_T présentée en chapitre E). Dans le but d'obtenir un tel planificateur nous proposons d'étudier les algorithmes permettant de retrouver le plan optimal, ainsi que l'ensemble des plans consistants. La présentation de ces deux algorithmes permettra d'introduire l'algorithme *P4OS* proposant un plan solution et optimal lorsque celui-ci existe.

G.4.1 Évaluation du plan optimal

Étant donnée une matrice de matching (ou matrice de liens sémantiques), nous présentons l'algorithme permettant de retrouver le poids du plan optimal au sens de la fonction de similarité Sim_T. Nous introduisons cet algorithme dans le but de retrouver le plan optimal mais non nécessairement solution pour notre problème de planification. Cet algorithme évalue donc le poids du plan optimal au moyen de la matrice de matching présentée lors du chapitre F. Afin d'éliminer les problèmes de boucles[5] nous introduisons un liste VB permettant de traiter les répétitions de buts (voir figure G.4). Cette liste a pour effet de renseigner sur les buts en cours de résolution sur une même branche de l'arbre de planification. Ainsi si un but a déjà été

[5]Nous ferons l'analogie entre le problème des boucles d'un problème de composition automatique de services web et l'anomalie de Sussman[147, 178] pour un problème de planification.

rencontré, alors l'algorithme est renseigné sur l'existence d'une boucle ce qui permet de rendre l'algorithme déterministe.

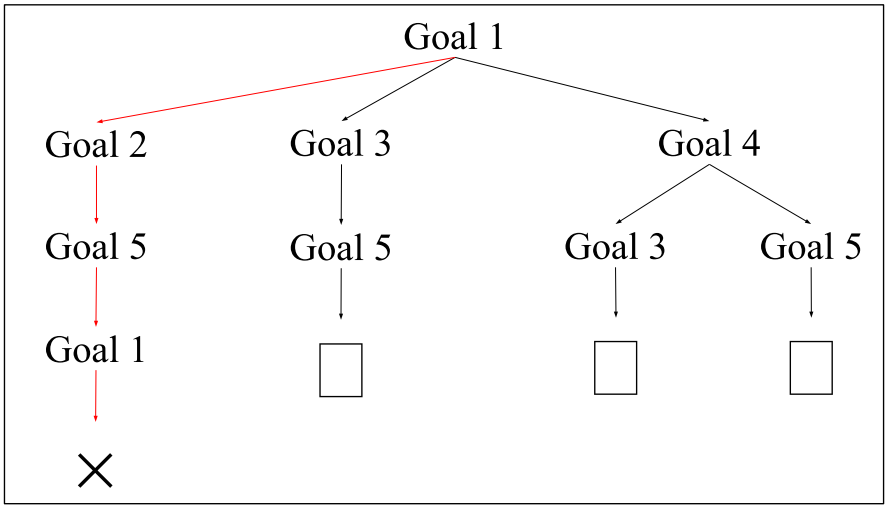


Figure G.4: Détection de boucles pour notre problème de planification.

Algorithm 10: Evaluation du plan optimal: *Eval*

Données: un ensemble de services Web: $S_{WebServices} = \{Ws_1, ..., Ws_n\}$,
la matrice de liens sémantiques $\mathscr{M}$,
la base de connaissance $\mathcal{KB}$,
l'ensemble des buts locaux β_L,
un liste de buts résolus VB.

Résultat: le poids du plan optimal.

début
  si $\beta \in \mathcal{KB}$ **alors**
    poids $\leftarrow 1$;
    supprimer (β, β_L);
  sinon
    $S_t \leftarrow \emptyset$;
    //Découverte des services resolvant le but β.
    pour chaque $i \in \bigcup_{i=1}^{n} pre(Ws_i)$ **faire**
      si $\mathscr{M}(i,\ \beta) \neq (\emptyset,\ 0)$ **alors**
        si $\mathscr{M}(i,\ \beta).set \notin S_t$ **alors**
          Ajouter$(\mathscr{M}(i,\ \beta).set, S_t)$;
        fin
      fin
    fin
    si $S_t \neq \emptyset$ **alors**
      si $\beta \in VB$ **alors**
        poids $\leftarrow poids * 0$;
      sinon
        Ajouter(β, VB);
        poids
        $\leftarrow Max_{S_t}(\frac{1}{\#pre_S^2} \sum_{pre_S} \mathscr{M}(pre_j, \beta).score * (\sum_{pre_S}(Eval(pre_j,\ VB))))$;
      fin
    sinon
      poids $\leftarrow poids * 0$;
    fin
  fin
fin

1. **Preuve d'arrêt:**

   L'algorithme suivant est constitué d'une unique boucle ainsi que de plusieurs tests conditionnels. Tout d'abord le test d'appartenance de β à la base de connaissance s'effectue en temps fini puisque la base de connaissance est assimilée à un ensemble fini par définition. La suppression d'un élément dans un ensemble fini et l'affectation sont deux opérations s'effectuant en temps fini.

   Notons que l'unique boucle permettant la *"Découverte des services résolvant le but β"* s'effectue sur l'ensemble des pré-conditions de services web qui est apparenté à un ensemble fini (la preuve a été faite dans le chapitre F lors de la construction de la matrice $\mathcal{M}$). La boucle précédente contient une imbrication de tests. Le premier test est une différence,

qui est une opération s'effectuant en temps fini. Le deuxième test est assimilé à un test d'appartenance, s'effectuant aussi en temps fini, tout comme la fonction *"Ajouter"*.

Enfin l'opération principale fait appel de nouveau à la fonction *Eval*. Cette fonction est donc assimilé à une fonction récursive, itérant sur les pré-conditions de services appartenant à S_t. L'arrêt de la fonction récursive est assurée puisque le cas difficile des boucles est contrôlé par la liste VB (liste des buts déjà rencontrés au cours du parcours en profondeur). De plus si le but appartient à la base de connaissance, alors le poids est assimilé à 1. Si l'ensemble S_t est vide alors le poids est initialisé a 0, tout comme dans le cas où le but à traiter appartient à VB. Ainsi comme l'ensemble des services web et leurs pré-conditions sont des ensembles finis, et comme le cas des boucles sont contrôlés alors la fonction de récurrence s'effectue en temps fini.

2. **Preuve de correction:**

   L'algorithme 10 a pour but de retourner le poids du plan solution optimal. Montrons que cet algorithme produit cet effet.

   Soit l'hypothèse de récurrence (HR) suivante:

$$Poids_{Max} = Max_{S_t}(\frac{1}{\#pre_S^2}\sum_{pre_S}\mathscr{M}(pre_j,\beta).score * (\sum_{pre_S}(Eval(pre_j,\ VB))))$$

   Nous procéderons par récurrence sur l'ensemble des services présents dans S_t. L'étape d'initialisation consiste à vérifier (HR) pour un ensemble S_t vide. Ainsi, nous supposons $S_t = \emptyset$ d'après l'algorithme 10. Le poids du plan optimal est donc égal à zéro ce qui est conforme à l'hypothèse de récurrence puisque aucun service ne peut satisfaire le but à résoudre.

   Nous supposons S_t un ensemble à $n+1$ éléments. De plus nous supposons que l'hypothèse de récurrence est vérifiée pour les n premiers éléments. Ainsi il existe un plan solution optimal vérifiant l'hypothèse de récurrence (HR). Donc il existe $s_x \in S_t$ tel que s_x fasse partie du plan solution optimal. Si nous supposons notre $n+1^{ieme}$ élément s faisant partie de S_t alors celui-ci fera partie du plan si et seulement si

$$\frac{1}{\#pre_s^2}\sum_{pre_s}\mathscr{M}(pre_j,\beta).score * (\sum_{pre_s}(Eval(pre_j,\ VB))) > \frac{1}{\#pre_{s_x}^2}\sum_{pre_{s_x}}\mathscr{M}(pre_j,\beta).score * (\sum_{pre_{s_x}}(Eval(pre_j,\ VB))) \quad \text{(G.1)}$$

   Ainsi le $poids_{max}$ du plan solution sera dépendant de s et non de s_x. Dans le cas contraire le $poids_{max}$ du plan solution sera dépendant de s_x et non de s. La fonction étudiée étant une fonction récursive, le processus est itéré jusqu'à qu'il n'y ait plus de pré-conditions à résoudre. Cependant des cas de boucles peuvent se produire si nous n'incluons pas le test $\beta \in VB$. En effet dans le cas de non présence d'un tel test, le processus peut boucler sur une pré-condition qui cherche à être satisfaite et qui n'appartient pas à $\mathcal{KB}$.

3. **Analyse de la complexité:**

   Nous supposons que le coût algorithmique d'un test conditionnel tel l'égalité, et l'appartenance s'effectue en temps constant. La découverte des services résolvant le but β est effectuée au sain d'un boucle itérant sur le nombre de pré-conditions de l'ensemble des services

$S_{WebServices}$[6]. Ainsi cette opération s'effectue en $\theta(p)$. Dans le pire des cas si l'on suppose n services ayant chacun p pré-conditions et q effets et si l'on suppose m le nombre maximal d'itérations alors la complexité de l'algorithme est en $\theta((np)^m)$, d'où $\theta(\exp^{m \ln(np)})$ donc exponentiel dans des cas non rencontrés en pratique puisqu'ils relatent de cas inconsistants.

[6]Par définition $p = Card(\bigcup_{i=1}^{n} pre(Ws_i))$.

G.4.2 Construction de plans consistants

Algorithm 11: Construction de plans consistants (CPC).

Données: un ensemble de services Web: $S_{WebServices} = \{Ws_1, ..., Ws_n\}$,
la matrice de liens sémantiques $\mathscr{M}$,
la base de connaissance $\mathcal{KB}$,
l'ensemble des buts locaux β_L,
une variable locale de sous-buts en cours de traitement VB,
un plan initialisé à $\emptyset$.

Résultat: l'ensemble des plans consistants.

début
  si $\beta \in \mathcal{KB}$ **alors**
    Concaténation(plan, (β));
  sinon
    $S_t \leftarrow \emptyset$;
    //Découverte des services resolvant le but β.
    pour chaque $i \in \bigcup_{i=1}^{n} pre(Ws_i)$ **faire**
      si $\mathscr{M}(i,\ \beta) \neq (\emptyset,\ 0)$ **alors**
        si $\mathscr{M}(i,\ \beta).set \notin S_t$ **alors**
          Ajouter($\mathscr{M}(i,\ \beta).set, S_t$);
        fin
      fin
    fin
    si $S_t \neq \emptyset$ **alors**
      pour chaque $s \in S_t$ **faire**
        plan $\leftarrow$ Concaténation(plan, $(\bigvee_{S_t} s \succ)$);
        pour chaque *pre(s)* **faire**
          si $\beta \in VB$ **alors**
            Concaténation(plan, $\emptyset$);
            Ajouter(VB, β_{LNV});
          sinon
            Ajouter(β, VB);
            plan $\leftarrow$ Concaténation(plan,$(\bigwedge_{pre(s)}$ CPC(plan, pre(s), VB)));
          fin
        fin
      fin
    sinon
      Concaténation(plan, $\emptyset$);
    fin
  fin
  retourner plan;
fin

L'algorithme 11 permet la construction de l'ensemble des plans consistants pouvant résoudre notre problème de planification. Cependant les plans proposés ne sont pas nécessairement solution puisqu'ils peuvent ne pas être corrects et/ou complets. La correction et la complétude

des plans sont assurées par l'algorithme général *P4OS*, qui vérifie les bonnes propriétés d'un plan solution. Tout comme dans l'algorithme 10 nous prenons en compte les cas de boucles ne rendant pas déterministe la recherche de plans consistants au moyen d'une liste de buts en cours de résolution VB.

Définition 18. *(Priorité des opérateurs)*
Étant donné un formalisme de description de problèmes de planification, nous avons l'ordre de priorité sur les opérateurs suivant:

$$\wedge > \vee > \succ$$

$\wedge$ *représente la multiplicité des paramètres d'entrées pour un service donné;*
$\vee$ *représente la multiplicité des services pour un but ou sous but unique;*
$a \succ b$ *si et seulement si les effets de* b *satisfont les pré-conditions de* a*.*

1. **Preuve d'arrêt:**

   L'algorithme 11 est constitué d'une unique boucle ainsi que de plusieurs tests conditionnels, tout comme l'algorithme d'évaluation du plan solution optimal. Comme pour les autres algorithmes, nous supposons que les tests d'appartenance et d'égalité s'effectuent en temps fini. De plus l'ensemble $\mathcal{KB}$ est un ensemble fini, donc le parcours de cet ensemble fini s'effectue en temps fini. La découverte des services résolvant le but β s'effectue en temps fini (voir preuve d'arrêt de l'algorithme 10). Ainsi l'algorithme 11 s'arrête si et seulement si les opérations effectuées à partir du test $S_t \neq \emptyset$ s'effectuent en temps fini. L'ensemble S_t est fini par construction donc le parcours d'un tel ensemble s'effectue en temps fini. Il en est de même pour l'ensemble des pré-conditions des services de S_t qui est constitué d'un sous ensemble fini de services de $S_{WebServices}$. L'ajout et la concaténation sont des opérations s'effectuant en temps fini donc par conséquent s'arrêtent.

   Enfin l'opération principale fait appel de nouveau à la fonction CPC (Construction des plans consistants). Nous sommes donc dans le cas d'une fonction récursive nécessitant des instructions d'arrêt. Les deux principales conditions d'arrêt portent sur l'appartenance du sous but β à la base de connaissance. La deuxième condition d'arrêt est rencontrée lorsque que S_t est assimilé à l'ensemble vide. Cependant, afin de ne pas boucler indéfiniment, il est nécessaire de s'assurer de la réduction du nombre d'appels de la fonction CPC. Cette réduction est assurée grâce au test d'appartenance de β à VB permettant de gérer les cas de boucles. Ce cas permet donc de réduire constamment le nombre d'appels de la fonction CPC puisque le nombre de services web est fini (tout comme le nombre de pré-conditions).

2. **Preuve de correction:**

   L'algorithme 11 a pour but de retourner l'ensemble des plans consistants. Montrons que cet algorithme produit cet effet.

   Traitons tous d'abord les deux cas triviaux:

   - si le but local à résoudre appartient à la base de connaissance alors celui-ci peut être resolu par hypothèse. De plus le plan proposé est consistant puisqu'il n'y a pas de contradictions lors de l'ordonnancement.
   - si le but local à résoudre ne peut être résolu par aucun des services ($S_t = \emptyset$) alors le plan est initialisé au plan vide qui est consistant par définition.

   Traitons le cas où le but local n'appartient pas à la base de connaissance et $S_t \neq \emptyset$:

- Ainsi il existe un service $s \in S_t$ ayant p pré-conditions telles que le plan résultant sera sous la forme: $P := s \succ CPC(p_1) \wedge ..., \wedge CPC(p_p)$[7]. Ainsi construire un tel plan consiste à analyser récursivement chacune des pré-conditions p_i comme un sous but. Le résultat final sera une disjonction de plans consistants.

La fonction étant une fonction récursive, le processus est itéré jusqu'à qu'il n'y ait plus de pré-conditions à résoudre. Cependant des cas de boucles peuvent se produire si nous n'incluons pas le test $\beta \in VB$. En effet dans le cas de non présence d'un tel test, le processus peut boucler sur une pré-condition qui cherche à être satisfaite et qui n'appartient pas à $\mathcal{KB}$ (Voir G.4). De plus la présence du test $\beta \in VB$ assure la consistance de l'ensemble des plans retournés.

3. **Analyse de la complexité: même complexité que l'algorithme 10**

   Nous supposons qu'un test conditionnel tel l'égalité, et l'appartenance s'effectue en temps constant. La découverte des services résolvant le but β est effectuée au sain d'un boucle itérant sur le nombre de pré-conditions de l'ensemble des services $S_{WebServices}$[8]. Ainsi cette opération s'effectue en $\theta(p)$. Dans le pire des cas si l'on suppose n services ayant chacun p pré-conditions et q effets et si l'on suppose m le nombre maximal d'itérations alors la complexité de l'algorithme est en $\theta((np)^m)$, d'où $\theta(\exp^{m \ln(np)})$ donc exponentiel dans des cas non rencontrés en pratique puisqu'ils relatent de cas inconsistants.

G.4.3 Construction du plan solution optimal: *P4OS*

L'algorithme *P4OS* permet la construction d'un plan solution optimal pour notre problème de planification. Il construit un plan solution optimal pour chacun des sous buts de β_L.

[7] Nous avons simplifié $CPC(P, p_i, VB)$ par $CPC(p_i)$.
[8] Par définition $p = Card(\bigcup_{i=1}^{n} pre(Ws_i))$.

Algorithm 12: Construction du plan solution optimal: *P4OS*.

Données: un ensemble de services Web: $S_{WebServices} = \{Ws_1, ..., Ws_n\}$,
la matrice de liens sémantiques $\mathscr{M}$,
la base de connaissance $\mathcal{KB}$,
l'ensemble des buts locaux β_L.

Résultat: le plan solution optimal.

début
  plan $\leftarrow \emptyset_P$;
  pour chaque *but local* $\beta \in \beta_L$ **faire**
    $plan_\alpha \leftarrow$ (ConstruirePlan($\emptyset$, β, $\emptyset$));
    plan_possible $\leftarrow$ $plan_\alpha$.Eval($plan_\alpha$);
    tant que $\exists\emptyset \in$ *plan_possible* **faire**
      $plan_\alpha \leftarrow plan_\alpha \backslash plan\_possible$;
      si $plan_\alpha \neq \emptyset_P$ **alors**
        plan_possible $\leftarrow$ $plan_\alpha$.Eval($plan_\alpha$);
      sinon
        Pas de plan solution optimal;
      fin
    fin
    //Construction d'une conjonction de plans indépendants.
    plan $\leftarrow$ ET(plan, plan_possible);
  fin
  retourner plan;
fin

Le plan final est une conjonction (ou ordre partiel) de plans solutions optimaux. De plus l'algorithme 12 possède la propriété de planificateur complet et optimal (voir propriété "Planificateur complet et optimal").

Définition 19. *(Objectif et plans indépendants) Deux sous objectifs G_1 et G_2 sont indépendants si deux plans P_1 et P_2 peuvent être calculés indépendamment l'un de l'autre pour réaliser G_1 et G_2 respectivement, et en exécutant les deux plans dans n'importe quel ordre, par exemple P_1 suivi de P_2. Nous réalisons $G1 \wedge G2$ par le plan $P_1 \wedge P_2$.*

1. **Preuve d'arrêt:**

   L'instanciation du plan au plan vide s'effectue en temps fini. L'arrêt de la première boucle (*Parcours de la liste des buts locaux*) de l'algorithme 12 est assurée puisque β_L est un ensemble fini de buts locaux. Par conséquent, l'analyse d'un ensemble fini s'arrête en temps fini.

   Par construction, $plan_\alpha$ représente un ensemble fini de plans dont chacun contient un nombre fini d'éléments $\emptyset$. Ainsi la boucle sur les éléments $\emptyset$ des éléments de $plan_\alpha$ s'effectue en temps fini. La suppression d'un plan *plan_possible* à un ensemble de plan $plan_\alpha$ est une opération s'effectuant en temps constant et par conséquent en temps fini. Ainsi cet algorithme s'effectue en temps fini puisque les fonctions *Eval* et *ConstruirePlan* s'arrêtent au bout d'un temps fini.

2. **Preuve de correction:**

   Supposons la fonction *Eval* retournant le plan ayant un poids maximal au sens de notre métrique Sim_T introduit lors du chapitre E. De plus considérons $ConstruirePlan$ la fonction permettant de construire un ensemble de plan consistants (non nécessairement solutions) en fonction de la matrice de Matching $\mathscr{M}$ (ou matrice de liens sémantiques) et d'un but β à résoudre. L'algorithme propose de retourner le plan solution optimal permettant la composition de services web. Ainsi le plan retourné est par définition:

   i) complet;

   ii) correct;

   iii) consistant;

   iv) optimal.

   Montrons que l'algorithme proposé retourne un plan complet, correct, consistant, et optimal.

   Tout d'abord supposons que l'ensemble des buts locaux soit instancié à l'ensemble vide, ainsi la liste de buts locaux β_L est nulle. Le problème de planification se schématise sous la forme $< S_{WebServices}, \mathcal{KB}, \emptyset >$. Le plan solution à ce problème est le plan $<>$ qui est complet, correct, consistant, et optimal d'après la propriété 12. Comme le plan a été initialisé au plan vide alors l'algorithme retourne le plan vide qui est un plan solution et optimal.

   Si l'on suppose qu'il existe au moins un élément β dans la liste de buts locaux β_L, alors $plan_\alpha$ est instancié à un ensemble de plans au moyen de la fonction $ConstruirePlan$. Par définition, la fonction $ConstruirePlan$ construit des plans consistants au problème de planification $< S_{WebServices}, \mathcal{KB}, \beta >$. Afin de proposer un unique plan optimal au problème de planification, l'ensemble des plans consistants $plan_\alpha$ est élagué par la fonction *Eval* qui ne retourne que le plan optimal au problème de planification $< S_{WebServices}, \mathcal{KB}, \beta >$. Ainsi nous avons un plan consistant et optimal à notre problème de planification. Cependant le plan proposé n'est pas nécessairement solution. En effet certains sous buts du plan proposé peuvent ne pas être résolus.

   Un plan possédant un élément $\emptyset$ signifie l'existence d'un sous but ne pouvant être résolu. Ainsi il existe un service web s présent dans le plan tel qu'il existe une de ses pré-conditions non résolue. Un tel plan n'est donc pas complet et non correct. Afin d'éliminer les plans non corrects et complets, nous proposons d'éliminer les plans contenant des éléments nuls. Ainsi pour chaque plan consistant et optimal proposé, nous vérifions la présence d'un élément nul. Ainsi s'il n'existe aucun plan consistant et optimal ayant les propriétés de correction et complétude alors il n'existe pas de plan solution au problème de planification.

   Si l'on suppose que le plan *plan_possible* est un plan solution optimal au problème $< S_{WebServices}, \mathcal{KB}, \beta >$ avec $\beta \in \beta_L$, alors la solution finale du problème
   $< S_{WebServices}, \mathcal{KB}, \beta_L >$ est une conjonction de plans solutions optimaux indépendants.

   Ainsi si le problème de planification $< S_{WebServices}, \mathcal{KB}, \beta_L >$ possède un plan solution optimal alors celui-ci est retrouvé par l'algorithme 12.

3. **Analyse de la complexité:**

   L'initialisation de la liste $\mathcal{L}$ s'effectue en temps linéaire en fonction de la taille de β_L. Ainsi la complexité de l'instanciation de la liste $\mathcal{L}$ est de l'ordre de $\theta(Card(\beta_L))$.

L'initialisation de la matrice de Matching $\mathscr{M}$ s'effectue en temps quadratique en fonction de la taille de p et q. Ainsi la complexité de l'initialisation de $\mathscr{M}$ est de l'ordre de $\theta(pq)$. La complexité du processus d'instanciation et d'initialisation est donc dans le pire des cas quadratique ou de l'ordre $\theta(pq)$. Cependant, les algorithmes 10 et 11 étant exponentiels, il en est de même pour celui-ci. Nous insistons sur le fait que le *"Pire cas"* est en pratique inconsistant. Ainsi les algorithmes 10, 11 et 12 obtiennent des résultats corrects en pratique.

Propriété 14. *L'algorithme 12 retourne un plan solution optimal résolvant la composition automatique de services web.*

Proof. Par définition un plan est un plan solution si et seulement si celui-ci est complet, consistant, correct et optimal.

1) Complétude: l'algorithme 12 retourne un plan complet puisqu'il élimine les plans non complets de l'espace des plans solutions (voir preuve de correction de l'algorithme 12).

2) Consistance: l'algorithme 12 appelle la fonction *ConstruirePlan* retournant des plans consistants au problème de planification.

3) Correction: d'après le processus de construction du plan, tous les sous buts sont résolus. En effet le plan proposé est consistant. En particulier les buts appartenant à β_L sont résolus, ainsi le plan proposé est correct puisqu'il satisfait tout les buts de β_L à l'aide de la base de connaissance $\mathcal{KB}$.

4) Optimalité: l'algorithme 12 appelle la fonction *Eval* retournant des plans optimaux au problème de planification.

□

Remarque:
Le plan solution proposé par l'algorithme 12 est représenté par un conjonction de plans indépendants résolvant des buts indépendants. En effet chaque but local de la liste des buts locaux est résolu indépendamment des autres buts. Nous parlerons de plans indépendants ou plans avec ordre partiel.

Propriété 15. *(Planificateur complet et optimal)*
L'algorithme 12 possède un rôle de planificateur complet et optimal.

Proof. Par définition l'algorithme 12 propose un plan solution (complet, consistant et correct) lorsque celui-ci existe (voir preuve de correction). De plus le plan solution proposé est optimal (voir preuve de correction). Ainsi l'algorithme propose un plan complet et optimal lorsque celui-ci existe. Ainsi l'algorithme 12 remplie les conditions nécessaires d'un planificateur complet et optimal. □

G.5 Exemple

G.5.1 Contexte

Considérons l'exemple traité depuis le début de ce document. Nous considérons donc l'ensemble de services web pertinents $S_{WebServices}$ auquel nous ajoutons un service s_8 (afin d'obtenir un plan non solution dans l'ensemble des plans consistants) tel que:

- s_8(in Cinema, out Movie).

Ainsi l'ensemble de services web pertinents est assimilé à l'ensemble suivant:

$$S_{WebServices} = \{s_1,\ s_2,\ s_3,\ s_4,\ s_5,\ s_6,\ s_7,\ s_8\}$$

Nous supposons la même base de connaissance $\mathcal{KB}$. Ayant ajouté un service web s_8, nous devons calculer la nouvelle matrice de Matching $\mathscr{M}$ mise à jour (voir tableau G.3) à l'aide du tableau G.2.

Sim_T	s_5	s_6	s_7	s_8
$Ws_X(\mathcal{KB},\ Movie)$	0.6	0.5	0.3	0.9

Table G.2: Résultats nécessaires pour le cas $e \notin \beta_L$.

	Dir	Movie	UP	GA	MG	PN	FM	Year	Cinema
Dir	$(\mathcal{KB},1)$	$\{(s_5,0.6)$ $(s_6,0.5)\}$	$(\mathcal{KB},1)$	$(\emptyset,0)$	$(\mathcal{KB},1)$	$(\mathcal{KB},1)$	$(\emptyset,0)$	$(\mathcal{KB},1)$	$(\emptyset,0)$
Movie	$(\mathcal{KB},1)$	$(\emptyset,0)$	$(\mathcal{KB},1)$	$(\emptyset,0)$	$(\mathcal{KB},1)$	$(\mathcal{KB},1)$	$(\emptyset,0)$	$(\mathcal{KB},1)$	$(s_1,0.9)$
UP	$(\mathcal{KB},1)$	$(\emptyset,0)$	$(\mathcal{KB},1)$	$(\emptyset,0)$	$(\mathcal{KB},1)$	$(\mathcal{KB},1)$	$(\emptyset,0)$	$(\mathcal{KB},1)$	$(s_1,0.9)$
GA	$(\mathcal{KB},1)$	$(\emptyset,0)$	$(\mathcal{KB},1)$	$(\emptyset,0)$	$(\mathcal{KB},1)$	$(\mathcal{KB},1)$	$(\emptyset,0)$	$(\mathcal{KB},1)$	$\{(s_2,0.7)$ $(s_3,0.6)\}$
MG	$(\mathcal{KB},1)$	$(\emptyset,0)$	$(\mathcal{KB},1)$	$(\emptyset,0)$	$(\mathcal{KB},1)$	$(\mathcal{KB},1)$	$(\emptyset,0)$	$(\mathcal{KB},1)$	$(s_2,0.7)$
PN	$(\mathcal{KB},1)$	$(\emptyset,0)$	$(\mathcal{KB},1)$	$(s_4,0.8)$	$(\mathcal{KB},1)$	$(\mathcal{KB},1)$	$(\emptyset,0)$	$(\mathcal{KB},1)$	$(\emptyset,0)$
FM	$(\mathcal{KB},1)$	$(\emptyset,0)$	$(\mathcal{KB},1)$	$(\emptyset,0)$	$(\mathcal{KB},1)$	$(\mathcal{KB},1)$	$(\emptyset,0)$	$(\mathcal{KB},1)$	$(s_3,0.6)$
Year	$(\mathcal{KB},1)$	$(s_7,0.3)$	$(\mathcal{KB},1)$	$(\emptyset,0)$	$(\mathcal{KB},1)$	$(\mathcal{KB},1)$	$(s_7,0.5)$	$(\mathcal{KB},1)$	$(\emptyset,0)$
Cinema	$(\mathcal{KB},1)$	$(s_8,0.9)$	$(\mathcal{KB},1)$	$(\emptyset,0)$	$(\mathcal{KB},1)$	$(\mathcal{KB},1)$	$(\emptyset,0)$	$(\mathcal{KB},1)$	$(\emptyset,0)$

Table G.3: Matrice $\mathscr{M}$ finale.

G.5.2 Modélisation du problème

Étant donnée la matrice des liens sémantiques $\mathscr{M}$, nous utiliserons l'algorithme présenté en section précédente permettant de construire la composition optimale de services web au sens de la fonction de similitude Sim_T. Nous résoudrons donc le problème de planification suivant: $< S_{WebServices},\ \mathcal{KB},\ \beta_G >$ où $\beta_G = Cinema$.

Recherchons le plan solution optimal résolvant ce problème de planification à l'aide des algorithmes présentés en section précédente.

G.5.3 Recherche du plan optimal

Nous décrivons les résultats obtenus par les trois algorithmes afin d'expliquer le fonction de l'algorithme général.

L'algorithme 11 produit un ensemble de plans consistants décrits en figure G.5.

Si l'on développe l'ensemble des plans, nous obtenons six plans distincts décrits en figure G.6.

Ainsi l'algorithme 11 retrouve six plans consistants pour notre problème de planification. L'algorithme *Eval* permet d'évaluer le poids de chacun des plans afin de classer les plans en

$$\begin{array}{ll} \text{Plan} \equiv & s_1 \succ (UP \wedge (s_5 \succ D \vee s_6 \succ D \vee s_7 \succ Y \vee s_8 \succ \emptyset)) \\ \vee & s_2 \succ (MG \wedge (s_4 \succ PN)) \\ \vee & s_3 \succ (s_2 \succ (MG \wedge s_4 \succ PN)) \wedge (s_7 \succ Y) \end{array}$$

Figure G.5: Plans consistants retournés par l'algorithme 11.

$$\begin{array}{lll} \text{Plan} \equiv & s_1 \succ (UP \wedge (s_5 \succ D)) & (P_1) \\ \vee & s_1 \succ (UP \wedge (s_6 \succ D)) & (P_2) \\ \vee & s_1 \succ (UP \wedge (s_7 \succ Y)) & (P_3) \\ \vee & s_1 \succ (UP \wedge (s_8 \succ \emptyset)) & (P_4) \\ \vee & s_2 \succ (MG \wedge (s_4 \succ PN)) & (P_5) \\ \vee & s_3 \succ (s_2 \succ (MG \wedge s_4 \succ PN)) \wedge (s_7 \succ Y) & (P_6) \end{array}$$

Figure G.6: Plans consistants développés.

fonction de leur poids (nous rappelons que le poids est calculé en fonction du calcul de similarité Sim_T entre services chaînés). L'algorithme *Eval* permet donc d'évaluer chacun des plans proposés par l'algorithme 11.

Ainsi nous obtenons:

- $Eval(Plan) = Poids(P_1) = \frac{1}{2^2}(0.9 + 0.9)(1 + \frac{0.6}{1} * 1) = 0.72$
- $Eval(Plan \backslash P_1) = Poids(P_2) = \frac{1}{2^2}(0.9 + 0.9)(1 + \frac{0.5}{1} * 1) = 0.67$
- $Eval(Plan \backslash (P_1 \cup P_2)) = Poids(P_5) = \frac{1}{2^2}(0.7 + 0.7)(1 + \frac{0.8}{1} * 1) = 0.63$
- $Eval(Plan \backslash (P_1 \cup P_2 \cup P_5)) = Poids(P_3) = \frac{1}{2^2}(0.9 + 0.9)(1 + \frac{0.3}{1} * 1) = 0.58$
- $Eval(Plan \backslash (P_1 \cup P_2 \cup P_5 \cup P_3)) = Poids(P_4) = \frac{1}{2^2}(0.9 + 0.9)(1 + \frac{0.9}{1} * 0) = 0.45$
- $Eval(Plan \backslash (P_1 \cup P_2 \cup P_5 \cup P_3 \cup P_4)) = Poids(P_6) = \frac{1}{2^2}(0.6 + 0.6)((\frac{0.8}{1} * 1) + (\frac{0.5}{1} * 1)) = 0.39$

L'algorithme général a pour but de retourner le plan optimal solution. En l'espèce la plan solution optimal de notre problème de planification est P_1. Il est clairement optimal puisque ce plan a été retrouvé par l'algorithme *Eval* (algorithme 10). De plus celui-ci est consistant par construction (algorithme 11), correct et complet d'après l'algorithme 12.

G.6 Synthèse

Lors de ce chapitre nous avons présenté un algorithme ayant un rôle de planificateur complet et optimal. Ainsi par définition, s'il existe un plan solution optimal résolvant le problème de planification, alors l'algorithme présenté retourne ce plan. Nous notons que les problèmes critiques de boucles (assimilés à l'anomalie de Sussman[9]) ont été traités ainsi l'algorithme proposé est

[9] Anomalie de Sussman: un plan pour atteindre un sous but est défait pour atteindre un autre sous but .

déterministe.

Améliorations:

Dans le but d'optimiser l'algorithme, il serait intéressant de prendre en compte les buts déjà résolus βLV. Ainsi le processus de planification n'est pas de nouveau exécuté si celui-ci appartient à l'ensemble des buts résolus. De même, il serait intéressant de prendre en compte une liste de buts non resolvables βLNV permettant d'élaguer l'espace des plans tentant de résoudre des sous buts n'ayant pas de solution.

Bibliography

[1] Samson Abramsky. Proofs as processes. *Theor. Comput. Sci.*, 135(1):5–9, 1994.

[2] Sudhir Agarwal, Siegfried Handschuh, and Steffen Staab. Annotation, composition and invocation of semantic web services. *J. Web Sem*, 2(1):31–48, 2004.

[3] R. Akkiraju, J. Farrell, J.Miller, M. Nagarajan, M. M. Schmidt, Amit Sheth, and Kunal Verma. Web service semantics - wsdl-s. Technical report, LSDIS Lab, 2005.

[4] Rama Akkiraju, Biplav Srivastava, Anca-Andreea Ivan, Richard Goodwin, and Tanveer Fathima Syeda-Mahmood. Semaplan: Combining planning with semantic matching to achieve web service composition. In *ICWS*, pages 37–44, 2006.

[5] Rama Akkiraju, Biplav Srivastava, Anca-Andreea Ivan, Richard Goodwin, and Tanveer Fathima Syeda-Mahmood. Semaplan: Combining planning with semantic matching to achieve web service composition. In *AAAI*, 2006.

[6] Rama Akkiraju, Biplav Srivastava, Anca-Andreea Ivan, Richard Goodwin, and Tanveer Fathima Syeda-Mahmood. Semaplan: Combining planning with semantic matching to achieve web service composition. In *AAAI*, 2006.

[7] Luca De Alfaro and Thomas A. Henzinger. Interface automata. In *ESEC / SIGSOFT FSE*, pages 109–120, 2001.

[8] Gustavo Alonso, Fabio Casati, Harumi Kuno, and Vijay Machiraju. *Web Services: Concepts, Architectures and Applictions*. Springer-Verlag, 2004.

[9] J. L. Ambite and J. Blythe. Icaps 2004 workshop on planning and scheduling for web and grid services (p4wgs 2004),http://www.isi.edu/ikcap/icaps04-workshop/. 2004.

[10] T. Andrews, F. Curbera, H. Dholakia, Y. Goland, J. Klein, F. Leymann, K. Liu, D. Roller, D. Smith, S. Thatte, I. Trickovic, and S. Weerawarana. Business process execution language for web services (version 1.1). Technical report, May 2004.

[11] Tony Andrews, Francisco Curbera, Hitesh Dholakia, Yaron Goland, Johannes Klein, Frank Leymann, Kevin Liu, Dieter Roller, Doug Smith, Satish Thatte, Ivana Trickovic, and Sanjiva Weerawarana. Business process execution language for web services, version 1.1. Technical report, OASIS, 2003-05-05 2003.

[12] Anupriya Ankolenkar, Massimo Paolucci, Naveen Srinivasan, and Katia Sycara. The owl services coalition, owl-s 1.1 beta release. Technical report, July 2004. http://www.daml.org/services/owl-s/1.1B/.

[13] Danilo Ardagna and Barbara Pernici. Adaptive service composition in flexible processes. *IEEE Trans. Software Eng.*, 33(6):369–384, 2007.

[14] Ismailcem Budak Arpinar, Boanerges Aleman-Meza, Ruoyan Zhang, and Angela Maduko. Ontology-driven web services composition platform. In *CEC*, pages 146–152, 2004.

[15] Ismailcem Budak Arpinar, Ruoyan Zhang, Boanerges Aleman-Meza, and Angela Maduko. Ontology-driven web services composition platform. *Inf. Syst. E-Business Management*, 3(2):175–199, 2005.

[16] Daniel Austin, Abbie Barbir, Christopher Ferris, and Sharad Garg. Web Services Architecture Requirements. Online: http://www.w3.org/TR/wsa-reqs/.

[17] F. Baader, R. Küsters, and R. Molitor. Computing least common subsumer in description logics with existential restrictions. In T. Dean editor, editor, *Proceedings of the 16th Int. Joint Conf. on AI*, 1999.

[18] Franz Baader. A graph-theoretic generalization of the least common subsumer and the most specific concept in the description logic el. In *WG*, pages 177–188, 2004.

[19] Franz Baader, Ralf Küsters, and Ralf Molitor. Rewriting concepts using terminologies. In *KR*, pages 297–308, 2000.

[20] Andrew Baker. *Matrix Groups: An Introduction to Lie Group Theory*. Springer undergraduate mathematics series. Springer-Verlag, London, 2002.

[21] Ajay Bansal, Srividya Kona, Luke Simon, and Thomas D. Hite. A universal service-semantics description language. In *ECOWS*, pages 214–225, 2005.

[22] Gianluigi Bellin and Philip J. Scott. On the pi-calculus and linear logic. *Theoretical Computer Science*, 135(1):11–65, 1994.

[23] B. Benatallah, M. Hacid, A. Leger, C. Rey, and F. Toumani. On automating web services discovery. *VLDB Journal*, pages 1–26, December 2002.

[24] Boualem Benatallah, Mohand-Said Hacid, Alain Léger, Christophe Rey, and Farouk Toumani. On automating web services discovery. *VLDB J.*, 14(1):84–96, 2005.

[25] D. Berardi, D. Calvanese, G. De Giacomo, M. Lenzerini, and M. Mecella. Automatic composition of e-services that export their behavior. In *1st Int. Conf. on Service Oriented Computing (ICSOC)*, pages 43–58, 2003. volume 2910.

[26] Daniela Berardi, Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, and Massimo Mecella. Automatic service composition based on behavioral descriptions. *Int. J. Cooperative Inf. Syst.*, 14(4):333–376, 2005.

[27] Daniela Berardi, Giuseppe De Giacomo, Maurizio Lenzerini, Massimo Mecella, and Diego Calvanese. Synthesis of underspecified composite -services based on automated reasoning. In *ICSOC*, pages 105–114, 2004.

[28] Rainer Berbner, Michael Spahn, Nicolas Repp, Oliver Heckmann, and Ralf Steinmetz. Heuristics for qos-aware web service composition. In *ICWS*, pages 72–82, 2006.

[29] Tim Berners-Lee, James Hendler, and Ora Lassila. The semantic web. *Scientific American*, 284(5):34–43, May 2001.

[30] Piergiorgio Bertoli, Jörg Hoffmann, Freddy Lécué, and Marco Pistore. Integrating discovery and automated composition: from semantic requirements to executable code. In *ICWS*, pages 815–822, 2007.

[31] Aysu Betin-Can, Tevfik Bultan, and Xiang Fu. Design for verification for asynchronously communicating web services. In *WWW*, pages 750–759, 2005.

[32] Warren Blanchet, Eleni Stroulia, and Renée Elio. Supporting adaptive web-service orchestration with an agent conversation framework. In *ICWS*, pages 541–549, 2005.

[33] J. Bloomer. *Power Programming with RPC*. 1992.

[34] D. Booth, H. Haas, F. McCabe, and et al. Web services architecture, w3c working group note 11. http://www.w3.org/TR/ws-arch/, 2004.

[35] S. Brandt, R. Kusters, and A. Turhan. Approximation and difference in description logics. In *KR*, pages 203–214, 2002.

[36] S. Brandt, R. Küsters, and A.-Y. Turhan. Approximation and difference in description logics. Technical Report Technical Report LTCS-Report 01-06, Aachen University of Technology, Research Group for Theoretical Computer Science, Germany, June 2001.

[37] T. Bray, J. Paoli, and et al. Extensible markup language, version 1.1. http://www.w3.org/TR/2004/REC-xml11-20040204/, 2004.

[38] Dan Brickley and R. V. Guha. Resource description framework (rdf) model and syntax specification. W3c recommendation, World Wide Web Consortium, February 1999.

[39] Dan Brickley and R. V. Guha. Rdf vocabulary description language 1.0: Rdf schema. W3c recommendation, World Wide Web Consortium, February 2004.

[40] Antonio Brogi, Sara Corfini, and Razvan Popescu. Composition-oriented service discovery. In *Software Composition, 4th International Workshop*, pages 15–30, 2005.

[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 Press.

[42] Schlenoff C., Gruninger M., Tissot F., Valois J., Lubell J., and Lee J. The Process Specification Language (PSL): Overview and version 1.0 specification. NISTIR 6459, National Institute of Standards and Technology, Gaithersburg, MD., 2000.

[43] Gerardo Canfora, Massimiliano Di Penta, Raffaele Esposito, and Maria Luisa Villani. An approach for qos-aware service composition based on genetic algorithms. In *GECCO*, pages 1069–1075, 2005.

[44] D. Caragea and T.F. Syeda-Mahmood. Semantic api matching for automatic service composition. In *WWW (Alternate Track Papers & Posters)*, pages 436–437, 2004. http://doi.acm.org/10.1145/1013513.

[45] J. Cardoso and A. Sheth. Semantic e-workflow composition. *Journal of Intelligent Information Systems*, 21:191–225, 2003.

[46] E. Christensen, F. Curbera, and et al. Web services description language, version 1.1. http://www.w3.org/TR/wsdl, 2001.

[47] William W. Cohen, Alexander Borgida, and Haym Hirsh. Computing least common subsumers in description logics. In *AAAI*, pages 754–760, 1992.

[48] S. Colucci, T. Di Noia, E. Di Sciascio, F.M. Donini, and M. Mongiello. A uniform tableaux-based approach to concept abduction and contraction in aln. In *DL-04 (2004), CEUR Workshop*, page 104.

[49] S. Colucci, T. Di Noia, E. Di Sciascio, F.M. Donini, and M. Mongiello. Concept abduction and contraction for semantic-based discovery of matches and negotiation spaces in an e-marketplace. In *Electronic Commerce Research and Applications*, volume 4, pages 41–50, 2005.

[50] Simona Colucci, Tommaso Di Noia, Eugenio Di Sciascio, Francesco M. Donini, and Marina Mongiello. Concept abduction and contraction in description logics. In *DL*, 2003.

[51] Ion Constantinescu, Boi Faltings, and Walter Binder. Type-based composition of information services in large scale environments. In *The 2004 IEEE/WIC/ACM International Conference on Web Intelligence (WI'04)*, September 2004.

[52] Ion Constantinescu, Boi Faltings, and Walter Binder. Type based service composition. In *WWW (Alternate Track Papers & Posters)*, pages 268–269, 2004.

[53] Ó. Corcho, A. Gómez-Pérez, A. Leger, C. Rey, and F. Toumani. An ontology-based mediation architecture for e-commerce applications. In *IIS*, pages 477–486, 2003.

[54] Marie Desjardins, Mithun Sheshagiri, and Timothy Finin. A planner for composing services described in DAML-S. In *ICAPS*, May 05 2003.

[55] Tommaso Di Noia, Eugenio Di Sciascio, Francesco M. Donini, and Marina Mongiello. Abductive matchmaking using description logics. In *Proceedings of Eighteenth IJCAI-03*, pages 337–342, Acapulco, Mexico, Aug. 2003. MK.

[56] T. Erl. *Service-Oriented Architecture: A Field Guide to Integrating XML and Web Services*. 2004.

[57] Jérôme Euzenat and Pavel Shvaiko. *Ontology matching*. Springer-Verlag, Heidelberg (DE), 2007.

[58] Baader F. and Nutt W. Basic description logics. In *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, 2003.

[59] J. Farrell and H. Lausen. Semantic annotations for wsdl and xml schema. Technical report, W3C Candidate Recommendation, January 2007. http://www.w3.org/TR/2007/CR-sawsdl-20070126/.

[60] Dieter Fensel, Michael Kifer, Jos de Bruijn, and John Domingue. Web service modeling ontology (wsmo) submission, w3c member submission. http://www.w3.org/Submission/WSMO/, June 2005.

[61] Donald F. Ferguson and Marcia L. Stockton. Service-oriented architecture: Programming model and product architecture. *IBM Systems Journal*, 44(4):753–780, 2005.

[62] R. Fielding, J. Gettys, J. Mogul, H. Frysyk, L. Masinter, P. Leach, and T. Berners-Lee. RFC 2616: Hypertext Transfer Protocol – HTTP/1.1. Technical report, IETF, 1999.

[63] Roy T. Fielding and Richard N. Taylor. Principled design of the modern web architecture. In *ICSE '00: Proceedings of the 22nd international conference on Software engineering*, pages 407–416, New York, NY, USA, 2000. ACM.

[64] R. Fikes and N. Nilsson. STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2:189–208, 1971.

[65] Agata Filipowska, Armin Haller, Monika Kaczmarek, Tammo van Lessen, Joerg Nitzsche, and Barry Norton. Super research deliverable D1.3: Process Ontology Language and Operational Semantics for Semantic Business Processes. Technical report, 2007.

[66] Howard Foster, Sebastián Uchitel, Jeff Magee, and Jeff Kramer. Compatibility verification for web service choreography. In *ICWS*, pages 738–741, 2004.

[67] I. R. Frank. *E. DCOM: Microsoft Distributed Component Object Model*. 1997.

[68] Xiang Fu, Tevfik Bultan, and Jianwen Su. Analysis of interacting bpel web services. In *WWW*, pages 621–630, 2004.

[69] Xiang Fu, Tevfik Bultan, and Jianwen Su. Analysis of interacting bpel web services. In *WWW*, pages 621–630, 2004.

[70] Raul Garcia-Castro and et al. Deliverable D1.2.4 (WP1.2) of European Union Project EU-IST-2004-507482 Knowledge Web: Architecture of the Semantic Web Framework v-1.0. Technical report, January 2007.

[71] Raul Garcia-Castro and et al. Deliverable D1.2.5 (WP1.2) of European Union Project EU-IST-2004-507482 Knowledge Web: Architecture of the Semantic Web Framework v-2.0. Technical report, December 2007.

[72] B.J Garner, D. Lukose, and E. Tsui. Parsing natural language through pattern correlation and modification. In *Proceedings of the 7th International Workshop on Expert Systems*, May 1987.

[73] Cagdas E. Gerede, Richard Hull, Oscar H. Ibarra, and Jianwen Su. Automated composition of e-services: lookaheads. In *ICSOC*, pages 252–262, New York, NY, USA, 2004. ACM Press.

[74] Cagdas Evren Gerede, Oscar H. Ibarra, Bala Ravikumar, and Jianwen Su. Online and minimum-cost ad hoc delegation in e-service composition. In *IEEE SCC*, pages 103–112, 2005.

[75] M. Ghallab, D. Nau, and P. Traverso. *Automated Planning: Theory and Practice*. Morgan Kaufmann Publishers, May 2004.

[76] Malik Ghallab et. al. PDDL-The Planning Domain Definition Language V. 2. Technical Report, report CVC TR-98-003/DCS TR-1165, Yale Center for Computational Vision and Control, 1998.

[77] G. De Giacomo and H.J. Levesque. An incremental interpreter for high-level programs with sensing. In H. Levesque and F. Pirri, editors, *Logical Foundations for Cognitive Agents, Contributions in Honor of Ray Reiter*, pages 86–102. Springer-Verlag, 1999.

[78] Giuseppe De Giacomo, Yves Lespérance, and Hector J. Levesque. Congolog, a concurrent programming language based on the situation calculus. *Artif. Intell.*, 121(1-2):109–169, 2000.

[79] Michael Gillmann, Gerhard Weikum, and Wolfgang Wonner. Workflow management with service quality guarantees. In *SIGMOD Conference*, pages 228–239, 2002.

[80] Javier Gonzalez-Castillo, David Trastour, and Claudio Bartolini. Description logics for matchmaking of services. In *Applications of Description Logics ADL*, 2002.

[81] M. Gudin, M. Hadley, and et al. Simple object access protocol. http://www.w3.org/TR/soap/, 2003.

[82] Volker Haarslev and Ralf Möller. RACER system description. *Lecture Notes in Computer Science*, 2083:701–??, 2001.

[83] David Harel and Amnon Naamad. The statemate semantics of statecharts. *ACM Trans. Softw. Eng. Methodol.*, 5(4):293–333, 1996.

[84] Ahlem Ben Hassine, Shigeo Matsubara, and Toru Ishida. A constraint-based approach to horizontal web service composition. In *ISWC*, pages 130–143, 2006.

[85] H.C.-Land and K. Yoon. Multiple criteria decision making. In *Economics and Mathematical Systems*, 1981.

[86] Abdelkrim Hebbar, Anna Zhdanova, Christian Rack, and Olaf Droegehorn. Spice research deliverable D4.1: Ontology Definition of User Profiles, Knowledge Information and Services. Technical report, 2006.

[87] J. Heinecke and F. Toumani. A natural language mediation system for e-commerce applications: an ontology-based approach. In *Proceedings of workshop on Human Language Technology for the Semantic Web and Web Services at ISWC*, 2003.

[88] C.A.R. Hoare. An axiomatic basis for computer programming. *Communications of the ACM*, 12(10):576–580, October 1969.

[89] Jörg Hoffmann, Piergiorgio Bertoli, and Marco Pistore. Web service composition as planning, revisited: In between background theories and initial state uncertainty. In *AAAI*, pages 1013–1018, 2007.

[90] Jörg Hoffmann and Ronen I. Brafman. Conformant planning via heuristic forward search: A new approach. *Artif. Intell.*, 170(6-7):507–541, 2006.

[91] I. Horrocks, P.F. Patel-Schneider, and F. van Harmelen. From SHIQ and RDF to OWL: The making of a web ontology language. *Journal of Web Semantics*, 2003.

[92] Ian Horrocks. Using an expressive description logic: FaCT or fiction? In *KR*, pages 636–649, 1998.

[93] Ian Horrocks, Peter F. Patel-Schneider, and Frank van Harmelen. Reviewing the design of daml+oil: An ontology language for the semantic web. In *AAAI/IAAI*, pages 792–797, 2002.

[94] Reiner Horst, Panos M. Pardalos, and Nguyen V. Thoai. *Introduction to global optimization.* Kluwer Academic Publishers, Dordrecht, 1995.

[95] Richard Hull, Michael Benedikt, Vassilis Christophides, and Jianwen Su. E-services: a look behind the curtain. In *PODS*, pages 1–14, 2003.

[96] H. Karloff. *Linear Programming*. Birkhauser, 1991.

[97] N. Kavantzas, D. Burdett, and G. Ritzinger. Web services choreography description language version 1.0. Working draft, W3C, http://www.w3.org/TR/2004/WD-ws-cdl-10-20040427, April 2004.

[98] Bartek Kiepuszewski, Arthur H. M. ter Hofstede, and Christoph Bussler. On structured workflow modelling. In *CAiSE*, pages 431–445, 2000.

[99] M. Klusch, B. Fries, M. Khalid, and K. Sycara. Owls-mx: Hybrid owl-s service matchmaking. In *AAAI Fall Symposium Series*, November 2005.

[100] Y. Kodratoff and G. Tecuci. Learning based on conceptual distance. In editor T. Dean, editor, *Proceedings of the 16th Int. Joint Conf. on AI*, 1988.

[101] Jacek Kopecký, Tomas Vitvar, Carine Bournez, and Joel Farrell. Sawsdl: Semantic annotations for wsdl and xml schema. *IEEE Internet Computing*, 11(6):60–67, 2007.

[102] Ralf Küsters. *Non-Standard Inferences in Description Logics*, volume 2100 of *Lecture Notes in Computer Science*. Springer, 2001.

[103] G. Lakemeyer. On sensing and off-line interpreting in golog. In H. Levesque and F. Pirri, editors, *Logical Foundations for Cognitive Agents, Contributions in Honor of Ray Reiter*, pages 173–187. Springer-Verlag, 1999.

[104] R. Lara, W. Binder, I. Constantinescu, D. Fensel, U. Keller, J. Pan, M. Pistore, A. Polleres, I. Toma, P. Traverso, and M. Zaremba. Knowledge Web research deliverable D2.4.2: Definition of semantics for web service discovery and composition. Technical report, 2004.

[105] O. Lassila and S. Dixit. Interleaving discovery and composition for simple workflows. In *Semantic Web Services, AAAI Spring Symposium Series*, pages 22–26, March 2004.

[106] M. Laukkanen and H. Helin. Composing workflows of semantic web services. In *Proceedings of the Workshop on Web-Services and Agent-based Engineering*, 2003. http://jmvidal.cse.sc.edu/library/laukkanen03a.pdf.

[107] Holger Lausen, Jos de Bruijn, Axel Polleres, and Dieter Fensel. Wsml - a language framework for semantic web services. In *W3C Rules Workshop*, Washington DC, USA, April 2005. Available from: http://www.w3.org/2004/12/rules-ws/paper/44/.

[108] Freddy Lécué and Alexandre Delteil. Making the difference in semantic web service composition. In *AAAI*, pages 1383–1388, 2007.

[109] Freddy Lécué, Alexandre Delteil, and Alain Léger. Applying abduction in semantic web service composition. In *ICWS*, pages 94–101, 2007.

[110] Freddy Lécué, Alexandre Delteil, and Alain Léger. Optimizing causal link based web service composition. In *Proceedings of the 18th European Conference on Artificial Intelligence*, pages ???–???, 2008.

[111] Freddy Lécué, Alexandre Delteil, and Alain Léger. Towards the composition of stateful and independent semantic web services. In *ACM SAC*, pages 2279–2285, 2008.

[112] Freddy Lécué, Olivier Boissier Alexandre Delteil, , and Alain Léger. Web service composition as a composition of valid and robust semantic links. *International Journal of Cooperative Information Systems (IJCIS)*, 17(4), December 2008.

[113] Freddy Lécué and Alain Léger. Causal link matrix and ai planning: a model for web service composition. In *Proceedings of the Workshop AI for Service Composition (ECAI)*, pages 66–68, August 2006.

[114] Freddy Lécué and Alain Léger. A formal model for semantic web service composition. In *ISWC the 5th International Semantic Web Conference*, pages 385–398, November 2006.

[115] Freddy Lécué and Alain Léger. A formal model for web service composition. In *ISPE CE*, pages 37–46, 2006.

[116] Freddy Lécué and Alain Léger. Semantic web service composition based on a closed world assumption. In *ECOWS*, pages 233–242, 2006.

[117] Freddy Lécué and Alain Léger. Semantic web service composition through a matchmaking of domain. In *ECOWS*, pages 171–180, 2006.

[118] Freddy Lécué, Aurélien Moreau, Samir Salibi, and Philippe Bron. Semantic and syntactic data flow in web service composition. In *Proceedings of the IEEE International Conference on Web Services (ICWS)*, pages 104–112, September 2008.

[119] Freddy Lécué, Eduardo Silva, Alain Pastor, and Maarten Steen. Deliverable D5.3.3 (WP5.3) of European Union Project EU-IST-2006-027617 Spice: Automated Composition Framework for Telecom Services. Technical report, may 2008.

[120] Freddy Lécué, Eduardo Silva, and Luís Ferreira Pires. A framework for dynamic web services composition. In *WEWST*, 2007.

[121] H. J. Levesque, R. Reiter, Y. Lesperance, F. Lin, and R. Scherl. GOLOG: A Logic programming language for dynamic domains. *Journal of Logic Programming*, 31(1-3):59–84, April-June 1997.

[122] Hector J. Levesque, Raymond Reiter, Yves Lespérance, Fangzhen Lin, and Richard B. Scherl. Golog: A logic programming language for dynamic domains. *J. Log. Program.*, 31(1-3):59–83, 1997.

[123] Frank Leymann. The (service) bus: Services penetrate everyday life. In *ICSOC*, pages 12–20, 2005.

[124] L. Li and I. Horrocks. A software framework for matchmaking based on semantic web technology. In *Proceedings of the Twelfth International Conference on World Wide Web*, pages 331–339. ACM Press, 2003.

[125] V. Lifschitz. On the semantics of strips. In *Reasoning about Actions and Plans*, pages 1–9, 1987.

[126] Moussa Lo and Fabien L. Gandon. Semantic web services in corporate memories. In *ICIW*, page 19, 2007.

[127] Sheshagiri M., DesJardins M., and Finin T. A planner for composing services described in daml-s. In *International Conference on Automated Planning and Scheduling (ICAPS). Workshop on planning for web services*, July 2003.

[128] Z. Morley Mao, Randy H. Katz, and Eric A. Brewer. Fault-tolerant, scalable, wide-area internet service composition. Technical report, October 25 2001.

[129] Annapaola Marconi, Marco Pistore, and Paolo Traverso. Implicit vs. explicit data-flow requirements in web service composition goals. In *ICSOC*, pages 459–464, 2006.

[130] Annapaola Marconi, Marco Pistore, and Paolo Traverso. Specifying data-flow requirements for the automated composition of web services. In *SEFM*, pages 147–156, 2006.

[131] David Martin, Massimo Paolucci, and Matthias Wagner. Bringing semantic annotations to web services: Owl-s from the sawsdl perspective. In *ISWC/ASWC*, pages 340–352, 2007.

[132] David McAllester and David Rosenblitt. Systematic nonlinear planning. pages 634–639, Menlo Park, CA, July 1991. AAAI.

[133] J. McCarthy and P.J. Hayes. Some philosophical problems from the standpoint of artificial intelligence. In M. Meltzer and D. Michie, editors, *Machine Intelligence 4*, pages 463–502. Edinburgh University Press, 1969.

[134] Drew V. McDermott. A heuristic estimator for means-ends analysis in planning. In *AIPS*, pages 142–149, 1996.

[135] Drew V. McDermott. Estimated-regression planning for interactions with web services. In *AIPS*, pages 204–211, 2002.

[136] D.V. McDermott. Pddl2.1 - the art of the possible? commentary on fox and long. *J. Artif. Intell. Res. (JAIR)*, 20:145–148, 2003.

[137] Deborah L. McGuinness and Peter F. Patel-Schneider. Usability issues in knowledge representation systems. In *AAAI/IAAI*, pages 608–614, 1998.

[138] Sheila A. McIlraith and Ronald Fadel. Planning with complex actions. In *NMR*, pages 356–364, 2002.

[139] Sheila A. McIlraith and Tran Cao Son. Adapting golog for composition of semantic web services. In *KR*, pages 482–496, 2002.

[140] Sheila A. McIlraith, Tran Cao Son, and Honglei Zeng. Semantic web services. volume 16, pages 46–53, 2001.

[141] Brahim Medjahed, Athman Bouguettaya, and Ahmed K. Elmagarmid. Composing web services on the semantic web. *VLDB J.*, 12(4):333–351, 2003.

[142] Microsoft Corperation and Digital Equipment Corperation. *The Component Object Model Specification (Version 0.9 ed.)*, 1995. Microsoft Corperation and Digital Equipment Corperation, One Microsoft Way, Redmond, WA: Microsoft Corperation and Digital Equipment Corperation.

[143] Marvin Minsky. A framework for representing knowledge. pages 163–189, 1995.

[144] S. Narayanan and S. McIlraith. Simulation, verification and automated composition of web services,. *Eleventh International World Wide Web Conference*, pages 7–10, May 2002.

[145] Dana S. Nau, Tsz-Chiu Au, Okhtay Ilghami, Ugur Kuter, J. William Murdock, Dan Wu, and Fusun Yaman. Shop2: An htn planning system. *J. Artif. Intell. Res. (JAIR)*, 20:379–404, 2003.

[146] A. Newell and H.A Simon. Gps, a program that simulates human thought. In *Computers and Thought*, pages 279–293. McGraw-Hill, New York, 1963.

[147] N.J. Nilsson. *Principles of artificial intelligence*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1980.

[148] OASIS SOA Reference Model Technical Committee. *Last accessed: 2nd March, 2008.*

[149] Object Management Group. *The Common Object Request Broker: Architecture and Specification*, 1998.

[150] J. OSullivan, D. Edmond, and A.t. Hofstede. What is in a service? *Distributed and Parallel Databases*, 12(2-3):117–133, September 2002.

[151] G. Oulsnam. Unravelling unstructured programs. *Comput. J.*, 25(3):379–387, 1982.

[152] M. Paolucci, T. Kawamura, T.R. Payne, and K. Sycara. Semantic matching of web services capabilities. In *Proceedings of the First International Semantic Web Conference, LNCS 2342*, pages 333–347. Springer-Verlag, june 2002.

[153] M. Paolucci, N. Srinivasan, and K. Sycara. Expressing wsmo mediators in owl-s. In *Semantic Web Services Workshop ,at the Third Interna-tional Semantic Web Conference*, 2004.

[154] Massimo Paolucci, Katia P. Sycara, and Takahiro Kawamura. Delivering semantic web services. In *WWW (Alternate Paper Tracks)*, 2003.

[155] Christos H. Papadimitriou. On the complexity of integer programming. *J. ACM*, 28(4):765–768, 1981.

[156] Mike P. Papazoglou. Service-oriented computing: Concepts, characteristics and directions. In *WISE*, pages 3–12, 2003.

[157] Mike P. Papazoglou. Web services technologies and standards. *ACM Computing Surveys*, pages 24–28, 2006.

[158] Mike P. Papazoglou and Dimitrios Georgakopoulos. Introduction to special issue on service-oriented computing. *Communications of ACM*, 46(10):24–28, 2003.

[159] Mike P. Papazoglou, Paolo Traverso, Schahram Dustdar, and Frank Leymann. Service-oriented computing: a research roadmap. *Int. J. Cooperative Inf. Syst.*, 17(2):223–255, 2008.

[160] Mike P. Papazoglou, Paolo Traverso, Schahram Dustdar, Frank Leymann, and Bernd J. Krämer. Service-oriented computing: A research roadmap. In *Service Oriented Computing*, 2005.

[161] Peter F. Patel-Schneider, Patrick Hayes, and Ian Horrocks. Owl web ontology language semantics and abstract syntax. W3c recommendation, World Wide Web Consortium, February 2004.

[162] Jyotishman Pathak, Samik Basu, Robyn R. Lutz, and Vasant Honavar. Parallel web service composition in moscoe: A choreography-based approach. In *ECOWS*, pages 3–12, 2006.

[163] Abhijit A. Patil, Swapna A. Oundhakar, Amit P. Sheth, and Kunal Verma. Meteor-s web service annotation framework. In *WWW*, pages 553–562, 2004.

[164] J. Penix and P. Alexander. Towards automated component adaptation. In *Proceedings of the 9th International Conference on Software Engineering and Knowledge Engineering*, June 1997.

[165] Minh Phan and Fumio Hattori. Automatic web service composition using congolog. In *ICDCS Workshops*, page 17, 2006.

[166] M. Pistore and P. Traverso. A theoretical integration of web service discovery and composition. Knowledge Web Deliverable D2.4.6A, 2006.

[167] Marco Pistore, Fabio Barbon, Piergiorgio Bertoli, D. Shaparau, and Paolo Traverso. Planning and monitoring web service composition. In *AIMSA*, pages 106–115, 2004.

[168] Marco Pistore, Piergiorgio Bertoli, Joerg Hoffmann, Freddy Lécué, David Portabella, and Radu Jurca. Deliverable D2.4.6.B (WP2.4) of European Union Project EU-IST-2004-507482 Knowledge Web: Report on Prototype for the Integration of Web Service Discovery and Composition. Technical report, August 2006.

[169] Marco Pistore, Annapaola Marconi, Piergiorgio Bertoli, and Paolo Traverso. Automated composition of web services by planning at the knowledge level. In *IJCAI*, pages 1252–1259, 2005.

[170] Marco Pistore, Pierluigi Roberti, and Paolo Traverso. Process-level composition of executable web services: "on-the-fly" versus "once-for-all" composition. In *ESWC*, pages 62–77, 2005.

[171] M. R. Quillian. Semantic memory. In editor M. Minsky, editor, *Semantic Information Processing*, pages 216–270. The MIT Press, 1968.

[172] R. Rada, H. Mill, E. Bicknell, and M.Bleumer. Development and application of a metric on semantic nets. In *IEEE Transactions on Systems, Man, and Cybernetics, Vol. 19, No.1*, Jan/Feb 1989.

[173] Azzurra Ragone, Tommaso Di Noia, Eugenio Di Sciascio, Francesco M. Donini, and Simona Colucci. Fully automated web services orchestration in a resource retrieval scenario. In *ICWS*, pages 427–434, 2005.

[174] Jinghai Rao, Peep Küngas, and Mihhail Matskin. Application of linear logic to web service composition. In *ICWS*, pages 3–9, 2003.

[175] Jinghai Rao, Peep Küngas, and Mihhail Matskin. Composition of semantic web services using linear logic theorem proving. *Inf. Syst.*, 31(4-5):340–360, 2006.

[176] Raymond Reiter. The frame problem in the situation calculus: a simple solution (sometimes) and a completeness result for goal regression. In Vladimir Lifschitz, editor, *Artificial Intelligence and Mathematical Theory of Computation*, pages 359–380, 1991.

[177] Raymond Reiter. *Knowledge in Action: Logical Foundations for Specifying and Implementing Dynamical Systems*. 2001.

[178] S.J. Russell and P. Norvig. *Artificial Intelligence: a modern approach*. Prentice-Hall, 1995.

[179] Battle S., Bernstein A., Boley A., Grosof B., Gruninger M., Hull R., Kifer M., Martin. D., McIlraith. S., McGuinness D., Su J., and Tabet S. Semantic web services framework (swsf). Technical report, W3C Member Submission, September 2005. http://www.w3.org/Submission/SWSF/.

[180] Ponnekanti S. and Fox A. Sword: A developer toolkit for web service composition. In *World Wide Web Conference*, pages 83–107, 2002.

[181] E.D. Sacerdoti. A structure for plans and behavior. In *Artificial Intelligence*, New York, 1977. Elsevier North-Holland.

[182] Sebastian Sardiña. Local conditional high-level robot programs. In *LPAR*, pages 110–124, 2001.

[183] R. Scherl and H. J. Levesque. Knowledge, action, and the frame problem. *Artificial Intelligence*, 144(1-2):1–39, 2003.

[184] Richard B. Scherl and Hector J. Levesque. The frame problem and knowledge-producing actions. In *AAAI*, pages 689–695, 1993.

[185] J. Schumann and B. Ficher. Nora/hammr: Making deduction-based software component retrieval practical. In *Proceedings of the 12th IEEE International Automated Software Engineering Conference (ASE97)*, November 1997.

[186] M.P. Singh and M.N. Huhns. *Service-Oriented Computing: Semantics, Processes, Agents*. John Wiley & Sons, 2005.

[187] Evren Sirin. *Combinning Description Logic Reasoning with AI Planning for Composition of Web Services*. PhD thesis, University of Maryland, College Park, 2006.

[188] Evren Sirin, James A. Hendler, and Bijan Parsia. Semi-automatic composition of web services using semantic descriptions. In *WSMAI*, pages 17–24, 2003.

[189] Evren Sirin and Bijan Parsia. Pellet: An OWL DL reasoner. In Volker Haarslev and Ralf Möller, editors, *Description Logics*, volume 104 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2004.

[190] Evren Sirin, Bijan Parsia, and James A. Hendler. Filtering and selecting semantic web services with interactive composition techniques. *IEEE Intelligent Systems*, 19(4):42–49, 2004.

[191] Evren Sirin, Bijan Parsia, Dan Wu, James Hendler, and Dana Nau. Htn planning for web service composition using shop2. *Web Semantics Journal*, pages 377–396, 2004.

[192] K. Sivashanmugam, K. Verma, A. Sheth, and J. Miller. Adding semantics to web services standards. In *the 1st International Conference on Web Services*, pages 395–401, 2003.

[193] Shirin Sohrabi, Nataliya Prokoshyna, and Sheila A. McIlraith. Web service composition via generic procedures and customizing user preferences. In *ISWC*, pages 597–611, 2006.

[194] Tanja Sollazzo, Siegfried Handschuh, Steffen Staab, Martin R. Frank, and Nenad Stojanovic. Semantic web service architecture – evolving web service standards toward the semantic web. In *FLAIRS Conference*, pages 425–429, 2002.

[195] Biplav Srivastava and Jana Koehler. Web service composition - current solutions and open problems. In *ICAPS*, July 22 2003.

[196] Vitaly A. Strusevich. Hans kellerer, ulrich pferschy and david pisinger, *napsack problems*, springer, berlin (2004) isbn 3-540-40286-1 546pp., eur 99, 95. *Oper. Res. Lett.*, 33(2):217–219, 2005.

[197] Katia P. Sycara, Massimo Paolucci, Anupriya Ankolekar, and Naveen Srinivasan. Automated discovery, interaction and composition of semantic web services. *J. Web Sem.*, 1(1):27–46, 2003.

[198] Gunnar Teege. Making the difference: A subtraction operation for description logics. In Jon Doyle, Erik Sandewall, and Pietro Torasso, editors, *KR*, pages 540–550, San Francisco, California, 1994. Morgan Kaufmann.

[199] Michael Thielscher. Inferring implicit state knowledge and plans with sensing actions. In *KI/ÖGAI*, pages 366–380, 2001.

[200] David Trastour, Claudio Bartolini, and Javier Gonzalez-Castillo. A semantic web approach to service description for matchmaking of services. In *SWWS*, pages 447–461, 2001.

[201] Paolo Traverso and Marco Pistore. Automated composition of semantic web services into executable processes. In *International Semantic Web Conference*, pages 380–394, 2004.

[202] Mark Turner, David Budgen, and Pearl Brereton. Turning software into a service. *IEEE Computer*, 36(10):38–44, 2003.

[203] S. Weerawarana, F. Curbera, F. Leymann, T. Storey, and D.F. Ferguson. *Web Services Platform Architecture*. 2005.

[204] L. Wolsey. *Integer Programming*. John Wiley and Sons, 1998.

[205] Dan Wu, Bijan Parsia, Evren Sirin, James A. Hendler, and Dana S. Nau. Automating DAML-S web services composition using SHOP2. In *International Semantic Web Conference*, pages 195–210, 2003.

[206] Tao Yu and Kwei-Jay Lin. Service selection algorithms for composing complex services with multiple qos constraints. In *ICSOC*, pages 130–143, 2005.

[207] A.M. Zaremski and J.M. Wing. Specification matching of software components. In *ACM Transactions on Software Engineering and Methodology*, 1997.

[208] Liangzhao Zeng, Boualem Benatallah, Marlon Dumas, Jayant Kalagnanam, and Quan Z. Sheng. Quality driven web services composition. In *WWW*, pages 411–421, 2003.

[209] Liangzhao Zeng, Boualem Benatallah, Anne H. H. Ngu, Marlon Dumas, Jayant Kalagnanam, and Henry Chang. Qos-aware middleware for web services composition. *IEEE Trans. Software Eng.*, 30(5):311–327, 2004.

[210] Ruoyan Zhang, Ismailcem Budak Arpinar, and Boanerges Aleman-Meza. Automatic composition of semantic web services. In *ICWS*, pages 38–41, 2003.

Ecole Nationale Supérieure des Mines
de Saint-Etienne

N° d'ordre : 492 I

Freddy Lécué
Titre de la thèse: Web Service Composition : Semantic Links based Approach
Spécialité: Computer Science

Mots clefs:
[Service Computing] Web Service, Service Composition, Composition Optimization;
[Artificial Intelligence] Semantic Web, Knowledge Representation, Automated Reasoning, AI Planning.

Résumé
Automated composition of Web services or the process of forming new value added Web services is one of the most promising challenges facing the Semantic Web today. Semantics enables Web service to describe capabilities together with their processes, hence one of the key elements for the automated composition of Web services. In this Ph.D study we focus on the functional level of Web services i.e., services are described according i) to some input, output parameters semantically enhanced by concepts in a domain ontology and ii) to preconditions and side-effects on the world. Web service composition is then viewed as a composition of semantic links controlled by causal laws. The semantic links refer to semantic matchmaking between Web service parameters (i.e., outputs and inputs) in order to model their connection and interaction whereas causal laws are the relationships between actions, action preconditions and side-effects. The key idea is that the matchmaking enables, at run time, finding semantic compatibilities among independently defined Web service descriptions. By considering such a level of composition we first study semantic links in details, and more specially their properties of validity and robustness.
From this and depending on services expressivity we focus on two different approaches to perform Web service composition. In the first approach a formal model to perform the automated composition of Web services by means of semantic links i.e., Semantic Link Matrix is introduced. This Semantic Link Matrix is required as a starting point to apply problem-solving techniques such as regression (or progression)-based search for Web service composition. The model supports a semantic context and focuses on semantic links in order to find correct, complete, consistent and robust plans as solutions. In this part an innovative and formal model for an Artificial Intelligence planning-oriented composition is presented.
In the second approach, besides semantic links, causal laws are also considered to achieve compositions of Web services. To this end an augmented and adapted version of the logic programming language Golog i.e., sslGolog is presented as a natural formalism not only for reasoning about the latter links and laws, but also for automatically composing services. sslGolog operates as an offline interpreter that supports n-ary output parameters of actions to compute conditional compositions of services. This approach is much more restrictive since assumes more expressivity on Web service description.
Finally, since Web services have been enhanced with formal semantic descriptions, the quality of semantic links involved in a composition is used as a innovative and distinguishing criterion to estimate its overall semantic quality. Therefore non functional criteria such as quality of service (QoS) are no longer considered as the only criteria to rank compositions satisfying the same goal.
In this part we focus on quality of semantic link based Web service composition. To this end, we present a general and extensible model to evaluate quality of both elementary and composition of semantic links. From this, we introduce a global semantic link selection based approach to compute the optimal composition. This problem is formulated as an optimization problem which is solved using efficient integer linear programming methods.
Our system is implemented and interacting with Web services dedicated on Telecommunication scenarios in use. The evaluation results showed high efficiency and effectiveness of the suggested approaches.

Ecole Nationale Supérieure des Mines
de Saint-Etienne

N° d'ordre : 492 I

Freddy Lécué

Titre de la thèse: Composition de Services Web: Une Approche basée liens sémantiques

Spécialité: Informatique

Mots clefs:

[Calcul basé Service] Service Web, Composition de Web services, Optimisation de compositions; [Intelligence Artificielle] Web Sémantique, Représentation des connaissances, Raisonnement Automatique, Planification en Intelligence Artificielle.

Résumé

La composition automatisée de services Web ou le processus de formation de nouveaux services Web à plus forte valeure ajoutée est l'un des plus grand défis auxquels le Web sémantique est face aujourd'hui. La sémantique permet d'un côte de décrire les capacités des services Web mais aussi leurs processus d'exécution, d'où un élément clé pour la composition automatique de services Web. Dans cette étude de doctorat, nous nous concentrons sur la description fonctionnelle des services Web c'est-à-dire, les services sont vus comme une fonction ayant des paramètres i) d'entrée, de sortie sémantiquement annotés par des concepts d'une ontologie de domaine et ii) des conditions préalables et effets conditionnels sur le monde. La composition de services Web est alors considérée comme une composition des liens sémantiques où les lois de cause à effets ont aussi un rôle prépondérant. L'idée maîtresse est que les liens sémantiques et les lois causales permettent, au moment de l'exécution, de trouver des compatibilités sémantiques, indépendamment des descriptions des services Web. En considérant un tel niveau de composition, nous étudions tout d'abord les liens sémantiques, et plus particulièrement leurs propriétés liées à la validité et la robustesse.

A partir de là et dépendant de l'expressivité des services Web, nous nous concentrons sur deux approches différentes pour effectuer la composition de services Web. Lors de la première approche, un modèle formel pour effectuer la composition automatique de services Web par le biais de liens sémantiques i.e., Matrice de liens sémantiques est introduite. Cette matrice est nécessaire comme point de départ pour appliquer des approches de recherche basées sur la régression (ou progression). Le modèle prend en charge un contexte sémantique et met l'accent sur les liens sémantiques afin de trouver des plans corrects, complets, cohérents et robustes comme solutions au problème de composition de services Web. Dans cette partie un modèle formel pour la planification et composition de services Web est présenté.

Dans la seconde approche, en plus de liens sémantiques, nous considérons les lois de causalité entre effets et pré-conditions de services Web pour obtenir les compositions valides de services Web. Pour ceci, une version étendue et adaptée du langage de programmation logique Golog (ici sslGolog) est présentée comme un formalisme naturel non seulement pour le raisonnement sur les liens sémantiques et les lois causales, mais aussi pour composer automatiquement les services Web. sslGolog fonctionne comme un interprète qui prend en charge les paramètres de sortie de services pour calculer les compositions conditionnelles de services. Cette approche (beaucoup plus restrictive) suppose plus d'expressivité sur la description de service Web.

Enfin, nous considérons la qualité des liens sémantiques impliqués dans la composition comme critère novateur et distinctif pour estimer la qualité sémantique des compositions calculées. Ainsi les critères non fonctionnels tels que la qualité de service(QoS) ne sont plus considérés comme les seuls critères permettant de classer les compositions satisfaisant le même objectif. Dans cette partie, nous nous concentrons sur la qualité des liens sémantiques appartenant à la composition de service Web. Pour ceci, nous présentons un modèle extensible permettant d'évaluer la qualité des liens sémantiques ainsi que leur composition. De ce fait, nous introduisons une approche fondée sur la sélection de liens sémantiques afin de calculer la composition optimale. Ce problème est formulé comme un problème d'optimisation qui est résolu à l'aide de la méthode par programmation linéaire entière.

Notre système est mis en œuvre et interagit avec des services Web portant sur de scénarios de télécommunications. Les résultats de l'évaluation a montré une grande efficacité des différentes approches proposées.