



UNIVERSITÉ DE CLAUDE BERNARD LYON 1
UNIVERSITÉ DE TUNIS

ÉCOLE DOCTORALE INFORMATIQUES ET MATHEMATIQUES DE LYON
INSTITUT SUPÉRIEUR DE GESTION DE TUNIS

THÈSE EN COTUTELLE

pour obtenir le titre de

Docteur en Informatique de **l'Université Claude Bernard Lyon 1**

et de **l'Université de Tunis**
Spécialité : INFORMATIQUE

Présentée et soutenue par **Soumaya AMDOUNI**

WEB SERVICES COMPOSITION IN UNCERTAIN ENVIRONMENTS

soutenue le 24 Septembre 2015

devant le jury composé de :

Rapporteurs:	Faiez GARGOURI	Professeur à l'université de Sfax Tunisie
	Abdelkader HAMEURLAIN	Professeur à luniversité Toulouse
Examinateurs:	Richard CHEBIER	Professeur à l'université de Pau
	Marinette SAVONNET	MCF HDR à l'université de Bourgogne
Directeur:	Djamal BENSLIMANE	Professeur à l'université Lyon1
Co-directeur:	Mahmoud BARHAMGI	Maître de conférence à l'université Lyon1
Directeur:	Rim FAIZ	Professeur à l'université de Carthage Tunisie

Acknowledgments

I wish to express my deep gratitude to my advisor Prof. Djamal Benslimane, for his excellent guidance, caring, and patienc. I thank him for his continuous support and encouragement despite the distance. I am also grateful to my co-advisor Prof. Mahmoud Barhamgi for his consistent supports and encouragements. His advice and cooperation are very helpful. I thank my advisor in Tunisia Prof. Rim Faiz for her help, advices and attention on my work.

I would also like to thank my parents, my sister and my brother for their continuous moral support and encouragement with their best wishes. Their love accompanies me wherever I go.

This dissertation would not have been possible without the support of my lovely husband, who has been there for me every step on the way. His love and continuous support contributed a lot to my success. A special thank to my little angel "Sajed".

Finally, I would like to thank my friends in Tunisia, at LIRIS Laboratory and Lyon 1 University.

Abstract:

In this thesis we focus on the data web services composition problem and study the impact of the uncertainty that may be associated with the output of a service on the service selection and composition processes. This work is motivated by the increasing number of application domains where data web services may return uncertain data, including the e-commerce, scientific data exploration, open web data, etc. We call such services that return uncertain data as uncertain services.

In this dissertation, we propose new models and techniques for the selection and the composition of uncertain data web services. Our techniques are based on well established fuzzy and probabilistic database theories and can handle the uncertainty efficiently. First, we proposed a composition model that takes into account the user preferences. In our model, user preferences are modelled as fuzzy constraints, and services are described with fuzzy constraints to better characterize their accessed data. The composition model features also a composition algebra that allows us to rank the returned results based on their relevance to user's preferences. Second, we proposed a probabilistic approach to model the uncertainty of the data returned by uncertain data services. Specifically, we extended the web service description standards (e.g., WSDL) to represent the outputs' probabilities. We also extended the service invocation process to take into account the uncertainty of input data. This extension is based on the possible worlds theory used in the probabilistic databases. We proposed also a set of probability-aware composition operators that are necessary to orchestrate uncertain data services. Since a composition may accept multiple orchestration plans and not all of them compute the correct probabilities of outputs, we defined a set of conditions to check if a plan is safe (i.e., computes the probabilities correctly) or not. We implemented our different techniques and applied them to the real-estate and e-commerce domains. We provide a performance study of our different composition techniques.

Keywords: *Data services, composition, fuzzy preferences, rank, uncertain service, probabilistic, orchestration, safe.*

Résumé court: Cette thèse porte sur la composition des services de données et l'étude de l'impact de l'incertitude qui peut être associée à leurs données accessibles sur le processus de composition et de sélection de service. En effet, dans un contexte tel que l'Internet, il est de plus en plus reconnu que les données et les services d'accès aux données sont sujettes à des valeurs d'incertitude tout en exigeant des techniques de gestion plus sophistiquées. Dans cette thèse, nous enrichissons la description sémantique des services Web afin de refléter l'incertitude, et nous proposons de nouveaux mécanismes et modèles pour la sélection et la composition des services. Nos mécanismes sont basés sur les ensembles flous et les théories probabilistes. Tout d'abord, nous étendons notre modélisation précédente basée sur les vues RDF afin d'inclure les contraintes floues qui caractérisent les données accédées par les services. Nous proposons une algèbre de composition qui permet de classer les résultats retournés en fonction de leur pertinence par rapport aux préférences de l'utilisateur. Notre algèbre proposée repose sur les fondations de bases de données floues. En outre, nous optons pour l'approche probabiliste pour modéliser l'incertitude des données renvoyées par les services incertains. Nous étendons la description du service Web standard pour représenter les probabilités de sortie. L'invocation des services est également étendue pour tenir compte de l'incertitude. Cette extension est basée sur la théorie des mondes possibles utilisée dans les bases de données probabiliste. Nous définissons un ensemble d'opérateurs de composition qui sont nécessaires pour orchestrer les services de données. Pour chaque composition, plusieurs plans d'orchestration peuvent être possibles mais qui sont pas tous corrects, donc, nous définissons un ensemble de conditions pour vérifier si le plan est correct (Safe) ou pas. Nous fournissons une implémentation de nos différentes techniques et les appliquer aux domaines de l'immobilier et du commerce électronique. Nous implémentons ces services et nous fournissons également une étude de la performance de notre prototype de composition.

Mots-clefs: *Les services de données, composition, préférences floues, classer, service incertain, probabiliste, orchestration, safe*

Résumé long de la thèse:

Au cours des dernières années, le Web a subi une transformation majeure, passant d'un web des données à un web de services. Ceci permet essentiellement les organisations d'offrir leurs services. Les services Web sont des applications logicielles modulaires autonomes qui sont conçus pour effectuer une tâche spécifique. Des exemples typiques comprennent les services de retour d'informations à l'utilisateur, tels que les services de prévision de la météo, ou des services altérant l'état du monde, tels que réservation ou achat de services en ligne, etc. En outre, l'utilisation de services Web est généralement au sein des applications d'entreprise et les actifs logiciels sur le Web. Une tendance récente est l'utilisation des services Web comme un moyen fiable pour la publication et le partage de données. Actuellement, de nombreuses organisations fournissent un accès basé sur les services via leurs données en mettant leurs bases de données derrière les services web en fournissant une méthode indépendante, interopérable et uniforme pour interagir avec les données. Ce type de services web est appelé les services de données (ou services d'accès de données). Cependant, depuis que l'Internet a une croissance exponentielle avec l'augmentation du nombre de sites, la gestion des données devient un enjeu majeur dans l'industrie de la technologie de l'information. L'incertitude et l'incomplétude sont deux caractéristiques communes de l'information que nous traitons dans notre vie quotidienne. La plupart des moteurs de recherche Web lorsqu'ils sont interrogés avec un mot clé, une série de pages web ainsi que leurs probabilités peuvent correspondre le mot fourni. Aujourd'hui, les informations qui nous entourent dans ce monde informatique virtuel sont souvent incertaines et imprécises. La gestion de l'information incertaine et imprécise a reçu une haute attention dans de nombreux domaines (commerce électronique, etc.). L'aspect de l'incertitude des données, en dépit de sa grande importance, n'a jamais été considéré dans la recherche autour des services web et leur composition. Cette thèse est parmi les premiers à aborder les problèmes de l'incertitude des données dans la communauté des services web, qui constitue l'une de ses nombreuses contributions originales. Elle introduit les concepts de préférences floues au sein des services web et des services web incertains et étale les fondements de base pour leur description sémantique. Nous proposons des extensions de méthodes de compositions de services DaaS dans un environnement caractérisé par une forte

présence d'incertitude. En effet, dans un contexte tel que l'Internet, il est de plus en plus reconnu que les données et donc les services d'accès aux données sont sujettes à des valeurs d'incertitude tout en exigeant des techniques de gestion plus sophistiquées. Dans cette thèse, nous enrichissons les annotations sémantiques de services Web afin de refléter cette dimension d'incertitude, puis nous avons proposé des mécanismes de composition de services appropriés. Ces mécanismes sont basés sur l'ensemble flou et les théories probabilistes pour tenir compte des différences d'interprétation des mondes possibles des données incertaines. Tout d'abord, nous avons présenté une approche pour composer les services web tout en tenant compte des préférences floues de l'utilisateur. Nous avons proposé un modèle pour les services de données basés sur des vues plus de RDF sur des ontologies de domaine. Le modèle sémantique permet de caractériser les préférences sous forme de contraintes floues. Ensuite, notre modéle sélectionne les services pertinents qui peuvent mieux satisfaire les préférences des utilisateurs, planifie leur ordre d'exécution et génère le plan d'orchestration qui répond mieux à la requête floue. Nous avons proposé une algèbre pour orchestrer les services de données sélectionnés. L'algèbre proposée classe les résultats retournés en fonction de leurs pertinences aux préférences floues. En outre, nous avons proposé une approche probabiliste pour modéliser l'incertitude des résultats retournés par un service incertain. Le modèle estime qu'un service de données incertain a une certaine sémantique et comportement que ces services peuvent retourner des résultats incertains. Nous avons proposé un modèle d'invocation qui permet l'invocation de services de données et avec une certaine incertitude entrée. Dans le premier cas, le processus d'invocation récupère les probabilités des sorties du service. Dans le second cas, le processus d'invocation calcule les probabilités de résultats renvoyés sur la base des probabilités renvoyées par le service et la probabilité de l'entrée. Ensuite, nous avons défini la sémantique de la composition de services incertains basée sur la théorie de mondes possibles et nous avons remarqué que cette théorie nécessite l'exploration de différentes combinaisons de mondes possibles pour évaluer la composition. Le calcul de mondes possibles après l'invocation de chaque service est inefficace que le nombre de ces mondes est exponentiel avec le nombre de lignes. Ainsi, nous avons opté pour l'approche extensionnelle et nous avons proposé une algèbre qui permet de calculer les probabilités des sorties. Ces prob-

abilités sont très importantes pour calculer les meilleurs résultats, l'évaluation de la qualité des résultats, la prise des bonnes décisions, etc. Ensuite, nous avons montré que les plans de composition ne peuvent pas tous calculer correctement les probabilités de sortie. Nous avons étudié à travers des exemples l'exactitude de ces plans d'orchestration dans deux cas: tuples indépendants et tuples BID. Pour ce fait, nous avons proposé un ensemble de conditions qui doivent être vérifiées pour obtenir des probabilités correctes pour ces deux cas. Enfin, nous avons mis en place le système pour évaluer notre approche pour la composition de service DaaS à l'incertitude. Nous avons également mené une analyse des performances sur un ensemble large de données afin d'évaluer l'efficacité de notre approche. Les résultats ont montré que notre système peut gérer des centaines de services Web DaaS dans un délai raisonnable, même avec le calcul de grades et de probabilités.

Contents

1 Introduction **1**

1.1 Context . . . 2
1.2 Research challenges . . . 4
1.3 Contributions . . . 5
1.4 Dissertation outline . . . 6

2 Background And Related Work On Web Services and Uncertainty **8**

2.1 Overview on Web Services and Data Web Services . . . 9
2.1.1 Web Services Definition . . . 9
2.1.2 Web Service Model . . . 10
2.1.3 Web Services Standards . . . 11
2.1.4 Data Web Services . . . 12
2.1.5 Web Services Composition . . . 13
2.2 Uncertainty Managing . . . 14
2.2.1 Fuzzy Preferences Processing . . . 14
2.2.2 Uncertain Data Management . . . 16
2.3 Related work . . . 24
2.3.1 Web services composition and ranking . . . 24
2.3.2 Uncertainty in web services composition . . . 26
2.3.3 Safe plans . . . 28
2.4 Conslusion . . . 28

3 A Preference-Aware Query Model for Data Services **30**

3.1 Introduction . . . 31
3.1.1 Motivating Example . . . 31
3.1.2 Challenges . . . 32
3.1.3 Contributions . . . 33
3.2 Preferences-Based Data Service Composition Model . . . 34
3.2.1 Preference Queries . . . 34

3.2.2 Data Services . . . 36
3.2.3 Query Rewriting . . . 38
3.3 A ranking-aware algebra for data services compositions . . . 41
3.3.1 Scalar Grade based Results Ranking Algebra . . . 42
3.3.2 Vector Grade based Results Ranking Algebra . . . 46
3.3.3 Approach Overview . . . 48
3.4 Conclusion . . . 49

4 Handling Uncertainty in Web Services Composition 51
4.1 Introduction . . . 52
4.1.1 Motivating Scenario . . . 52
4.1.2 Challenges . . . 54
4.1.3 Contributions . . . 55
4.2 Uncertain Data Services: A Probabilistic Model . . . 56
4.2.1 A description model for uncertain data services . . . 56
4.2.2 An Invocation Model For Uncertain Data Services . . . 58
4.3 Uncertain Data Service Composition Model . . . 59
4.3.1 Composition Semantics . . . 60
4.3.2 An Algebra for Uncertain Data Services Composition . . . 63
4.4 Block Independent Disjoint (BID) services . . . 64
4.4.1 BID-P-Service Definition . . . 65
4.4.2 BID-P-Service invocation . . . 66
4.4.3 BID-P-Service composition . . . 67
4.5 Conclusion . . . 68

5 Safe Plan 69
5.1 Introduction . . . 70
5.1.1 Motivating scenario . . . 70
5.1.2 Challenges . . . 71
5.1.3 Contribution . . . 72
5.2 Background . . . 72
5.3 Safe composition for data services with independent tuples . . . 74
5.3.1 Example . . . 74

5.3.2 Criteria for safe composition plans . . . 76
5.4 Safe composition for data services with BID tuples . . . 78
5.5 Conclusion . . . 79

6 Implementation and evaluation 80
6.1 Introduction . . . 81
6.2 Prototype . . . 81
6.2.1 Architecture . . . 81
6.2.2 Service Annotation . . . 83
6.2.3 Service Invocation . . . 85
6.2.4 Query formulation . . . 85
6.2.5 Query rewriting . . . 86
6.2.6 Composition Execution . . . 86
6.3 Implementation and experimental results . . . 88
6.3.1 Technical environment . . . 88
6.3.2 Preference-Aware Query Model . . . 89
6.3.3 Uncertainty in Web Services Composition . . . 93
6.3.4 Experimental results . . . 93
6.4 Conclusion . . . 96

7 Conclusion 97
7.1 Conclusions . . . 98
7.2 Future works . . . 99

A Appendix: Academic Achievements 102

Bibliography 103

List of Figures

2.1 The Web Service Model 11
2.2 Basic fuzzy set types 16
2.3 Probabilistic apartment database and possible worlds space 20

3.1 (a) a Graphical Representation of the Query, (b) the Associated Membership Functions, (c) the formulated query in SPARQL 35
3.2 RDF Parametrized Views - Examples. 36
3.3 Composition Plan 44
3.4 The intermediate and final results along with their grades 46
3.5 An overview of the proposed approach 48

4.1 Sample of results returned by S_4^p and the corresponding possible worlds 57
4.2 Semantics of the probability-aware service invocation 59
4.3 Composition and Execution of uncertain Services 62
4.4 Evaluation of $Project_p^p(Invoke(S_2^p, Invoke(S_3^p, \text{“}Lyon\text{”})))$ 64
4.5 (A)Sample of results returned by the BID-P service S_4^p (B)The possible worlds of $S_4^p <\text{”}'Lyon\text{”}'>$ 65
4.6 Composition and Execution of Uncertain BID Services 66
4.7 (A)Invocation of $S_3^p(\$a, ?c, ?j)$ and $S_2^p(\$c, ?p, ?pr)$(B)The interpretation of a composition (C)The composition Plan of Q_1 (D)The results 67

5.1 An Example of a composition 71
5.2 Composition Plan 75
5.3 BID Composition Plan 79

6.1 System architecture 82
6.2 A Portion of a WSDL File Annotated with RDF Views 83
6.3 An example of a SOAP message 84
6.4 Implementation of uncertain Invocation based on the JAX-WS API 86

6.5 The preference query formulator interface 89
6.6 The fuzzy term editing 90
6.7 The execution strategy panel 91
6.8 The scalar ranked results panel 92
6.9 The vector ranked results panel 92
6.10 Uncertain composition system interface 93
6.11 Performance results 94
6.12 Performance results: measuring execution time with and without grades calculation 95
6.13 Performance results: measuring execution time with and without probabilities calculation 96

List of Tables

3.1 Available Web Services 32
3.2 Mapping Table: the covered sub graphs by sample data services . . 40
3.3 Implemented norms and conorms 42
3.4 Vector Ranking results 47

4.1 Examples of Data Web Services 53

6.1 Data processing operations 87

Chapter 1

Introduction

Contents

1.1 Context 2
1.2 Research challenges 4
1.3 Contributions 5
1.4 Dissertation outline 6

1.1 Context

Over the last decade the Web has undergone a major transformation, changing from an environment for mere data sharing among individuals to an environment that also allows organizations across all spectra to offer their services and conduct their daily business. Modern enterprises worldwide have already moved their operations to the Internet by adopting the Web service technology [Alonso 2004] to provide an interoperable and programmatic interface to their internal systems. The Web Service framework embodies the paradigm of Service-Oriented Computing (SOC) [Papazoglou 2003], whereby software applications both within and outside the enterprise walls are encapsulated as services that can be executed, composed and coordinated in a loosely-coupled manner. Simply put, a Web service is a piece of software application whose interface and binding can be defined, described and discovered as XML artifacts [Curbera 2002], and is accessible via ubiquitous Web protocols and standard data formats such as HTTP, XML and SOAP. The XML-based standards around the Web service technology are the key contributor to the large adoption and deployment of Web services. Three key XML-based standards have been defined to support Web service deployment: SOAP [1], WSDL[2], and UDDI [3]. SOAP defines a communication protocol for Web services. WSDL enables service providers to describe their services. UDDI offers a registry service that allows advertisement and discovery of Web services.

Besides using Web services to provide access to corporate applications and software assets over the Web, a recent trend has been to use Web services as a reliable means for data publishing and sharing among organizations [Carey 2007]. Today, many enterprises provide a service based access to their data on the Web by putting their databases behind Web services, thereby providing a well-documented, platform (and data source) independent, interoperable and uniform method of interacting with their data. We call this type of Web services as *data Web Services*, where services correspond to calls (i.e. parameterized queries) over the data sources' schemas. This is as opposed to traditional Web services that provide access to

[1] Simple Object Access Protocol (SOAP), http://www.w3.org/TR/soap/
[2] Web Services Description Language(WSDL), http://www.w3.org/TR/wsdl
[3] Universal Description, Discovery, and Integration (UDDI) http://www.uddi.org/pubs/uddiv3.htm

corporate applications and which we call as SaaS Web services (Software-as-a-Service Web Services) in the hereafter.

While individual data Web services may provide interesting information alone, in most cases, users' queries require the invocation of several services. For instance, let us consider the following query: *"what are the tests performed in ABC Lab by patients who have been administered Glucophage in XWZ hospital?"* Let us assume that ABC Lab and XWZ hospital provide two data services S_{ABC} and S_{XYZ}, respectively: S_{ABC} returns the tests performed by a given patient in ABC Lab and S_{XYZ} returns the list of patients that have been administered a given drug in XWZ hospital. The execution of the above mentioned query involves the composition of S_{ABC} and S_{XYZ} services. Web service composition is a powerful solution for building value-added services on top of existing ones.

In our PhD dissertation, we focus on the composition of data Web services and study the impact of the uncertainty that may be associated with their accessed data on the service composition and selection processes. Uncertainty and incompleteness are two common characteristics of the information we deal with in our everyday life. For example, most of the Web search engines that we use on a daily basis return, when queried with some keyword, a set of Web pages along with their probabilities of matching the supplied keyword; the products that are returned from querying the e-commerce sites (e.g., *eBay.com*, *apartment.com*, *amazon.com*) that we use daily are often associated with imprecise and incomplete information (e.g., in their prices, locations, descriptions, etc.) and have uncertain character as they may not really match our formulated queries. In the same way, the data that are returned by a Web service may be uncertain. To the best of our knowledge, the data uncertainty aspect, despite its high importance, was never considered in the previous research works around Web services and their composition. Our thesis is among the first works to address the data uncertainty issues in the Web services research community. Specifically, we proposed to describe services with "fuzzy preferences" to better characterize their accessed data, and introduced the concept of "uncertain Web services" and laid down the basic foundations for their semantic description, selection, composition and execution.

1.2 Research challenges

We summarize below the challenges that need to be dealt with when devising advanced techniques for web services composition in uncertain environments.

- *Modeling web services with fuzzy constraints*: The description of a Web service (e.g., WSDL, OWL-S, etc.) should include some meta information to characterize the data accessed by the service. This would largely improve the service selection and composition processes by allowing to focus on the services that match the user's preferences.

- *Modeling the uncertainty of web services*: The uncertainty of the data returned by a Web service should be modeled to allow its consumer to interpret and use it correctly. The uncertainty model should be compatible with current service description standards (e.g., WSDL) as these are widely adopted by Web service development community. For example, the proposed model should be integrated in WSDL in a way that does not affect service consumers that are unaware of uncertainty.

- *Composition algebra*: The conventional services' composition model (i.e., the composition algebra and its implementations by different composition execution engines) should be extended to allow for computing the probabilities/grades of the composition's outputs to help users understand and interpret them correctly. An uncertain service should be compose-able with normal and uncertain services alike; i.e., a composition that is unaware of uncertainty should be able to use uncertain services without affecting its normal execution.

- *Web services orchestration*: Given a composition (i.e., a set of services whose composition can answer a query), different orchestrations may be possible. An orchestration defines an execution plan of a composition. In the context of uncertain data Web services, not all orchestrations compute the probabilities of its outputs correctly. The challenge is to find, for a given query, the composition plan that computes the correct probabilities for the outputs.

1.3 Contributions

In this dissertation, we enrich the semantic description of Web services to reflect their uncertainty, and propose new mechanisms and models for services selection and composition. Our mechanisms are based on fuzzy set and probabilistic theories and handle the uncertainty efficiently. We summarize bellow the major contributions of this thesis:

- *Web services annotated with fuzzy constraints*: We extended our previous modeling to Web data services as RDF Views [Barhamghi 2010] to include fuzzy constraints that characterize their accessed data. The constraints are expressed as intervals or fuzzy sets.

- *A ranking-aware algebra for services composition*: We proposed a composition algebra that allows us to rank the returned results based on their relevance to user's preferences. Our proposed algebra relies on fuzzy databases foundations [Dubois 1990].

- *A probabilistic model for uncertain data services*: We opted for a probabilistic approach to model the uncertainty of the data returned by uncertain services. We extended the web service description standard to represent the outputs' probabilities. We proposed an invocation model for the invocation of uncertain data services with certain and uncertain inputs. The invocation process retrieves the probabilities of the service's outputs and computes the probabilities of returned results based on the probabilities returned by the service and the probability of the input. Moreover, we defined the semantics of uncertain service composition based on the possible world theory [Bosc 2010]. We proposed a probability-aware composition algebra to compute the probabilities of the composition outputs.

- *Safe orchestration plan*: because not all composition plans compute correctly the output probabilities, we defined a set of conditions to check whether a composition plan computes correctly the outputs' probabilities.

- *Implementation and evaluation*: We implemented our different techniques

and models and applied them to the real-estate and e-commerce domains. We conducted a performance study of our composition framework.

1.4 Dissertation outline

The rest of this dissertation is organized as follows.

- In Chapter 2, we provide the necessary background for understanding our different proposals in the dissertation. First, we present the key concepts around the Web service technology. We then focus specifically on the area of preferences. We introduce the reader to modeling and querying uncertain data, top-k query processing and ranking queries on uncertain data. Finally, we review the related work that are most related to our approach. This aims to position our work with respect to existing ones.

- In Chapter 3, we proposed a declarative approach for composing Web data services on the fly. We proposed to model data services as RDF Views over domain ontologies to represent their semantics declaratively. Our semantic model allows characterizing the returned data using the fuzzy set theory. Our approach is based on the use of the query rewriting techniques to automate the composition process, and allows to rank-order the composition results based on the user preferences.

- In Chapter 4, we proposed a probabilistic approach for modeling uncertain data services for two cases: independent data and Block-Independent-Disjoint data. Specifically, we showed how the uncertainty associated with a data service can be modeled, and proposed a composition algebra that can compute the probabilities of the outputs.

- In Chapter 5, we studied through examples the safety of the orchestration plans in two cases: independent tuples and BID tuples. Moreover, we proposed a set of conditions that should be verified to obtain correct probabilities for these two cases.

- In Chapter 6, we provide an implementation of our different techniques and apply them to the real-estate and e-commerce domains. We implemented the services and we provided a performance study of our composition framework.

- In Chapter 7, we provide concluding remarks and discuss some possible directions for future research.

Chapter 2

Background And Related Work On Web Services and Uncertainty

Contents

2.1 Overview on Web Services and Data Web Services 9

2.1.1 Web Services Definition 9

2.1.2 Web Service Model 10

2.1.3 Web Services Standards 11

2.1.4 Data Web Services 12

2.1.5 Web Services Composition 13

2.2 Uncertainty Managing 14

2.2.1 Fuzzy Preferences Processing 14

2.2.2 Uncertain Data Management 16

2.3 Related work 24

2.3.1 Web services composition and ranking 24

2.3.2 Uncertainty in web services composition 26

2.3.3 Safe plans 28

2.4 Conslusion 28

In this chapter, we review some of the key concepts in the areas of Web services and uncertainty management. Specifically, we define the following concepts and topics: *web services*, *data web services*, *web services composition*, *fuzzy preferences processing*, *modeling and querying uncertain data*, *top-k query processing and ranking queries on uncertain data*. We also report some of the most recent research works in these same areas.

2.1 Overview on Web Services and Data Web Services

2.1.1 Web Services Definition

A variety of definitions about Web services are given in the literature. However, that proposed by the Word Wide Web Consortium (W3C[1]) is considered as reference: *"A Web service is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL: Web Services Description Language). Other systems interact with the Web service in a manner prescribed by its description using SOAP (Simple Object Access Protocol) messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards[2]"*.
This definition highlights the major technological and business benefits of Web services, namely:

- *Interoperability*: This is the most important benefit of Web services. Web services typically work outside of private networks, offering developers a non-proprietary route to their solutions. Web services developed are likely, therefore, to have a longer life-span, offering better return on investment of the developed Web service. Web services also let developers use their preferred programming languages. In addition, thanks to the use of standards-based communications methods, Web services are virtually platform-independent.

[1] http://www.w3.org/
[2] http://www.w3.org/TR/ws-arch/

- *Usability*: Web services allow the business logic of many different systems to be exposed over the Web. This gives users' applications the freedom to choose the Web services that they need. Instead of re-inventing the wheel for each client, users need only include additional application-specific business logic on the client-side. This allows to develop services and/or client-side code using the languages and tools that users want.

- *Reusability*: Web services provide not a component-based model of application development, but the closest thing possible to zero-coding deployment of such Web services. This makes it easy to reuse Web service components as appropriate in other Web services. It also makes it easy to deploy legacy code as a Web service.

- *Deployability*: Web services are deployed over standard Internet technologies. This makes it possible to deploy Web services even over the fire wall to servers running on the Internet on the other side of the globe. Also thanks to the use of proven community standards, underlying security is already built-in.

2.1.2 Web Service Model

The Web service model is based upon the interactions between three types of participants including service provider, service registry and service client. Interactions involve three basic operations: service publishing, finding and binding. Participants and operations act upon the Web service artifacts encompassing the service implementation and description. Figure 2.1 shows the different participants and the interaction among them. In a typical scenario, a service provider provides a network-accessible software module, i.e., an implementation of a Web service, defines a service description for the Web service and publishes it to a service registry so that the service client can find it. The service description contains information such as the inputs/outputs of the Web service, the address where the service is located and QoS. The service client queries the service registry and retrieves the service description. Then it uses the information in the service description to bind with the service provider and invoke the Web service implementation.

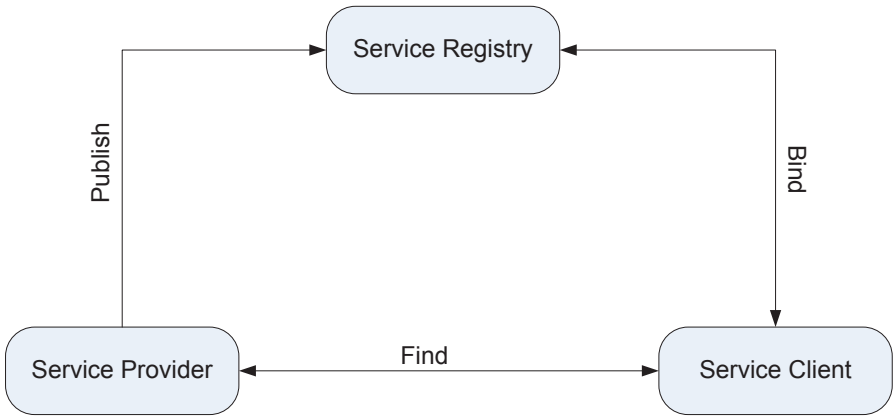


Figure 2.1: The Web Service Model

2.1.3 Web Services Standards

Standards are key enablers of Web services [Curbera 2002, Vaughan-Nichols 2002]. The service model from above is realized via the following XML-based standards:

- *Web Services Description Language(WSDL [3])*: WSDL is an XML-based language that is used for describing the functionality offered by a Web service. A WSDL description of a Web service provides a machine-readable description of how the service can be called, what parameters it expects, and what data structures it returns. A WSDL document describes a Web service at two levels: the *abstract* level and the *concrete* level. At the abstract level, the WSDL description includes three basic elements: **Type**, **Message**, and **PortType**. At the concrete level, the WSDL description provides information about binding.

- *Simple Object Access Protocol(SOAP [4])*: SOAP is a wrapper around the implementation of the OSI network model layer. The most used application protocol to transmit SOAP messages is HTTP, but it is also possible to use the SMTP or FTP protocols. A SOAP message contains one XML element (Envelope) and two child elements (Header and Body). The Envelope defines the namespaces for the remaining content of a SOAP message. The

[3] http://www.w3.org/TR/wsdl
[4] http://www.w3.org/TR/soap12

Header is an optional element. It can carry auxiliary information in a SOAP message. The Body is the mandatory part of a SOAP message. It specifies the information to be carried from the initial message sender to the ultimate message receiver.

- *Universal Description, Discovery and Integration(UDDI [5])*: UDDI is a platform independent, XML-based registry by which businesses worldwide can list themselves on the Internet, and a mechanism to register and locate web service applications. UDDI is an open industry initiative, sponsored by the Organization for the Advancement of Structured Information Standards (OASIS), for enabling businesses to publish service listings and discover each other, and to define how the services or software applications interact over the Internet.

2.1.4 Data Web Services

Besides using Web services to provide access to corporate applications and software assets over the Web, a recent trend has been to use Web services as a reliable means for data publishing and sharing among organizations [Carey 2007]. Today, many enterprises provide a service based access to their data on the Web by putting their databases behind Web services, thereby providing a well-documented, platform (and data source) independent, interoperable and uniform method of interacting with their data [Dan 2007]. We call this type of Web services as Data Web Services, where services correspond to calls over the data sources' schemas. A recently released report from Forrester Research [Gilpin 2007] has defined a Data Service as follows:
"An information service (i.e., data service) provides a simplified, integrated view of real-time, high-quality information about a specific business entity, such as a customer or product. It can be provided by middleware or packaged as an individual software component. The information that it provides comes from a diverse set of information resources, including operational systems, operational data stores, data warehouses, content repositories, collaboration stores, and even streaming sources

[5] http://www.uddi.org/pubs/uddi-v3.htm

in advanced cases".

Data as a Service brings the notion that data quality can happen in a centralized place, cleansing and enriching data and offering it to different systems, applications or users, irrespective of where they were in the organization or on the network. As such, Data as a Service solutions provide the following advantages:

- *Heterogeneity*: The adoption of data services and the SOA paradigm relieves SOA application developers from having to directly cope with the first two forms of heterogeneity. That is, in the world of Web services all data sources are described using WSDL and invoked via REST or SOAP calls (thus have the same interface), and all data are in XML form and described using XML Schema (thus have the same data model).

- *Agility*: the value-added of SOA to application development is reuse and agility, but without flexibility at the data tier, that value-added quickly erodes. Instead of relying on non-reusable proprietary codes to access and manipulate data in monolithic application silos, one can exploit data services that can be used and reused in multiple business processes. This greatly simplifies development and maintenance of service oriented applications, enforces compliant use of data, and introduces easy-to-use capabilities for using information in dynamic and real-time processes.

- *Data quality*: Access to data is controlled through the data services, which tends to improve data quality, as there is a single point for updates. Once those services are tested thoroughly, they only need to be regression tested, if they remain unchanged for the next deployment.

2.1.5 Web Services Composition

Web service composition is the process of selecting, combining and executing Web services (WS) in order to resolve user' requests that cannot be resolved based on individual services alone. A sheer number of research works were devoted to Web service composition over the last years [Eid 2008, Tabatabaei 2008, Weise 2008, Yu 2008]. Much of these works exploit the Semantic Web as a viable means for au-

tomating the composition process. Based on the involvement degree of users in the composition process and on the automation degree, WS composition can be conducted in three different fashions: manual (using some programming languages), semi-automatic (through a series of interactions with the user), and automatic.

2.2 Uncertainty Managing

Data Web services, and Web services in general, have received a considerable attention in the last few years [Yu 2008]. Previous research works [Yu 2008] have addressed the different aspects of the Web service lifecycle, including service creation, selection, composition, and execution. However, there are still many issues related to the quality of data Web services themselves that need to be explored and tackled [Carey 2007]. The *fuzzy* users' preferences and the *uncertainty* of the data returned by data Web services are one of the key issues that have not been yet fully explored.

2.2.1 Fuzzy Preferences Processing

The handling of user preferences is becoming an increasingly important issue in present-day information systems [Chomicki 2003]. Motivations for such a concern are manifold [Hadjali 2011]. First, it has appeared to be desirable to offer more expressive query languages which can be more faithful to what a user intends to say. Second, the introduction of preferences in queries provides a basis for rank ordering the retrieved items, which is especially valuable in case of large sets of items satisfying a query. Third, on the contrary, a classical query may also have an empty set of answers, while a relaxed (and thus less restrictive) version of the query might be matched by items in the database. User preferences are a key in ranking compositions' results and selecting the best ones. We give below some definitions.

Definitions. *A preference is an expression that represents a desire of the user over the attributes of a process model or activity [Abbaci 2011].*

User Preference is a concept which enables a choice between several objects and provides rank ordering of these objects, based on user' s satisfaction they provide. Therefore the simplest representation of user preference is object ordering or ranking [Gursky 2008].

Users' preferences can be modeled using fuzzy sets. Fuzzy sets theory [Zadeh 1965, Dubois 1996, de Calmes 2003, de Calmes 2007] is a flexible approach and present convenient tools to model vague criteria and user's preferences. Fuzzy sets are very well suited to the interpretation of linguistic terms and constitute a convenient way for users to express their preferences. Using this paradigm, a preference is represented by means of a set whose boundaries are gradual. Thus, the satisfaction of a tuple t regarding such a fuzzy set F is a matter of degree in the unit interval denoted by $\mu(t)$. The underlying fuzzy set theory offers a large panoply of connectives to aggregate these preferences from classical conjunction (min) and disjunction (max) to quantified statements (*most of*, *at least two*, *around a dozen*, ...) and weighted averaging operators. In the context of querying, users define fuzzy sets to model their preferences that are associated with linguistic labels like '*low*', '*very cheap*', etc. Moreover, in accordance with the imprecise nature of the concepts they represent, membership functions associated with the fuzzy sets behind these properties introduce some graduality when checking the satisfaction of the items. The satisfaction degree in $[0, 1]$ provides the necessary information to rank-order the items that somewhat satisfy the user's requirements.

The preferences usually belong to one of the following variants of trapezoidal membership function (figure 2.2):

- *Left-trapezoidal function*: lower attribute values are better. The satisfaction degree is computed as follows: $\mu(t) = \begin{cases} 1 & \text{if} \quad z \leq a \\ 0 & \text{if} \quad z \geq b \\ \frac{b-z}{b-a} & \text{if} \quad a < z < b \end{cases}$
- *Right-trapezoidal function*: higher attribute values are better. The satisfac-

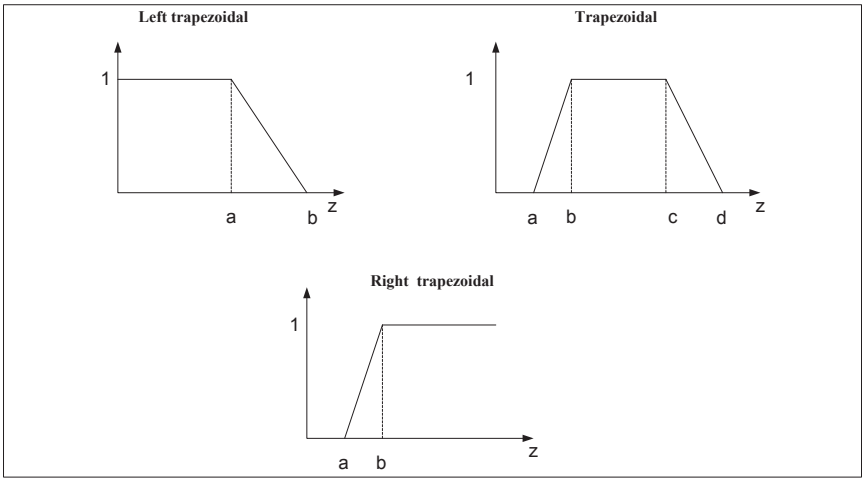


Figure 2.2: Basic fuzzy set types

tion degree is computed as follows: $\mu(t) = \begin{cases} 1 & \text{if} \quad z \geq b \\ 0 & \text{if} \quad z \leq a \\ \frac{b-z}{b-a} & \text{if} \quad a < z < b \end{cases}$

- *Trapezoidal function*: middle values are preferred. The satisfaction degree is computed as follows: $\mu(t) = \begin{cases} 1 & \text{if} \quad b \leq z \leq c \\ \frac{d-z}{d-c} & \text{if} \quad c < z < d \\ \frac{b-z}{b-a} & \text{if} \quad a < z < b \\ 0 & \text{Otherwise} \end{cases}$

2.2.2 Uncertain Data Management

Uncertainty and Incompleteness are two common characteristics of the information we deal with in our everyday life. Most of the Web search engines that we use on a daily basis return, when queried with some key words, a set of Web pages along with their probabilities of matching the supplied word; the information that we receive from the intelligent objects (e.g., *Smart phones, GPSs, ambient sensors, etc.*) that surround us in this pervasive computing world are often uncertain and imprecise. For example, the products that are returned from querying the e-commerce sites (e.g., eBay.com, apartment.com, amazon.com) that we use daily are often associated with imprecise and incomplete information (e.g., in their prices, locations, descriptions, etc.) and have uncertain character as they may not really

match our formulated queries. Managing such uncertain data is currently receiving increasing attention in many application domains, e.g., e-commerce, sensor networks [Tatbul 2004], scientific data exploration [Buneman 2006], data integration [Marian 2011, Agrawal 2010], moving objects tracking [Cheng 2004], data cleaning, information extraction, and location-based services, etc. For example, in the e-commerce domain, a recent study [Soliman 2010] showed that 65% of the business objects (e.g., apartments, cars, products, etc.) that one would find on business Web sites (e.g., apartments.com, carpages.ca, etc.) are associated with some uncertainty in their basic information (prices, locations, etc.). These domains exhibit uncertainty in their underlying data, coupled with increasing demand from users to efficiently derive high-quality answers for the queries posed on such data.

2.2.2.1 Data Uncertainty Types

Two types of uncertainty are used under the relational data model: *tuple level uncertainty* and *attribute level uncertainty*.

- *Tuple Level Uncertainty*: we do not know whether the tuple belongs to the database instance or not. The variable associated to the tuple has a Boolean domain: it is true when the tuple is present and false if it is absent. Such a tuple is also called a maybe tuple [Widom 2008]. A widely-used model to capture this type of uncertainty is representing tuples as probabilistic events, and model the database as a joint distribution defined on these events.

- *Attribute Level Uncertainty*: tuple attributes may have uncertain values. A widely-used model to capture this type of uncertainty is representing tuple attributes as probability distributions defined on discrete or continuous domains.
  We will find it convenient to convert attribute-level uncertainty into tuple-level uncertainty and consider only tuple-level uncertainty during query processing. This translation is done as follows. For every tuple t , where the attribute A takes possible values a_1, a_2, a_3, we create several tuples $t_1, t_2, t_3...$ that are identical to t except for the attribute A, whose values

are $t_1.A = a_1, t_2.A = a_2$ etc. Now each tuple t_i is uncertain and the tuples $t_1, t_2, ...$ are mutually exclusive.

2.2.2.2 Uncertain Data Models

Different approaches were proposed to model data uncertainty [Sadri 1991, Green 2006, Abiteboul 1987, Bosc 2010]. Among these models, the probabilistic and the possibilistic models are the most adopted due to their simplicity mainly the probabilistic approach. In the probabilistic data model, data uncertainty is modeled as a probability distribution over the possible tuple attribute values [Green 2006, Bosc 2010](each possible tuple/attribute value is assigned a degree of confidence, quantifying its probability).
The probabilistic model is a numerical model that relies on an additive assumption and adopts the Possible Worlds Semantics. The Possible Worlds [Bosc 2010] is an important concept for understanding the models of uncertain data. In the possible world semantics, data uncertainty is captured by viewing the database as a set of possible instances that correspond to the different possible instantiations of the uncertain data items. Many uncertainty models, e.g., [Abiteboul 1987, Imielinski 1984], adopt the possible worlds semantics, where a probabilistic database is viewed as a set of possible instances (worlds). The above concepts were the bases of several research projects and systems (e.g., TRIO [Widom 2005], ORION [Cheng 2007] , MystiQ [Dalvi 2007]...)that address modeling and querying uncertain data. The TRIO system introduced working models to capture uncertainty at different levels by relating uncertainty with lineage and leveraging existing DBMSs capabilities for uncertain data management. The ORION project deals with constantly evolving data in the form of continuous intervals, and presents query processing and indexing techniques for managing uncertain data in such representation. The possible worlds appraoch is defined as follows:

Definition. *Assume a relational schema with k relation names $R_1, ..., R_n$, a probabilistic database is a finite set of possible worlds $W = \{w_1, w_2, ..., w_k\}$ where each database instance $w_i = (R_1^i, ..., R_n^i)$.*

The possible worlds space represents an enumeration of all possible views of the database resulting from the uncertainty or incompleteness in the underlying data. Possible worlds probabilities are determined based on the probabilistic dependencies among tuples. The most commonly used tuple-level uncertainty model is the *independent tuples model* [Fuhr 1997]. This model associates existence probabilities with individual tuples and assumes that the tuples are independent of each other. On the other hand, if there are correlations between tuples in a database table, the table should be decomposed into simpler tables and we called them independent-disjoint tables.

Tuple Independent Model. A *tuple independent model* is a probabilistic model where all the tuples are probabilistic events. If the database consists of a single table, we refer to it a tuple independent table. The database is interpreted as a probability distribution over the set of all possible worlds [Halpern 1990]. Each world contains a subset of the tuples present in the probabilistic database and the probability of each world is calculated by multiplying the existence probabilities of present tuples and non-existence probabilities of non present tuples.

Block Independent Disjoint Model (BID). *A block independent disjoint model* is a probabilistic database where the set of tuples can be partitioned into blocks [Re 2007, Dalvi 2011, Stoyanovich 2011]. All tuples in a block are *disjoint* probabilistic events and all the tuples in different blocks are independent probabilistic events. The representation of a BID table is a folows: we choose the key attributes $a_1, a_2, ..., a_n$ of a relation T that uniquely identify the block to which tuple belongs, then we add a probability attribute P so the schema of a BID table is $T(a_1, ..., a_n, b_1, ..., b_k, P)$. The probability of a block is the sum of the membership probability values of all tuples in the block and the probability of each possible world is the joint event of the existence of world's blocks, and the absence of all other blocks.

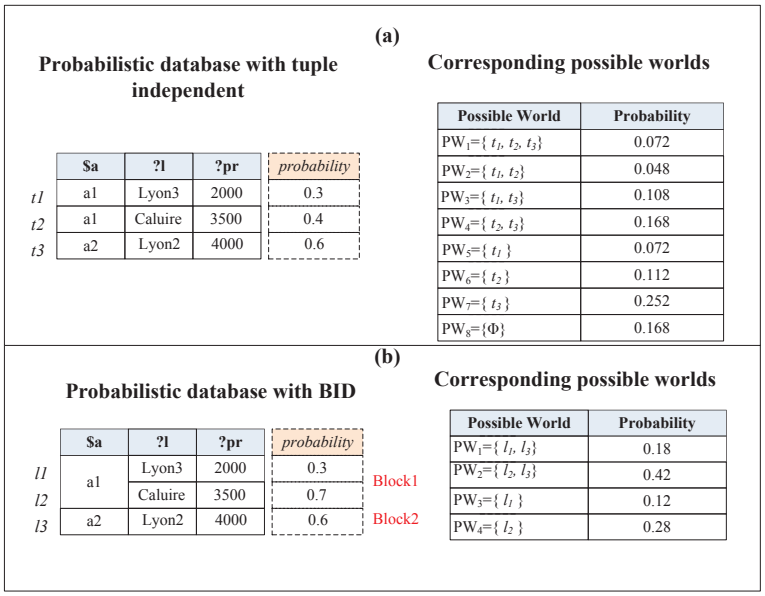


Figure 2.3: Probabilistic apartment database and possible worlds space

Example. Assume an apartment database reported by *apartments.com* for a simple search for available apartments to buy. Figure 2.3(a) shows the apartment table with tuple independent and the corresponding possible worlds, and their probabilities and Figure 2.3(b) shows block independent disjoint model of apartment table and its possible worlds. Each world can be seen as a joint event of the existence of world's tuples, and the absence of all other database tuples. The sum of probabilities of all possible worlds is equal to 1 and the probability in each model is calculated as follows:
Independent Tuple: $P(PW_5) = P(t_1)*(1-p(t_2))*(1-p(t_3)) = 0.3*0.6*0.4 = 0.072$
Block-independent-disjoint: $P(PW_4) = P(l_2) * (1 - p(l_3)) = 0.7 * 0.4 = 0.28$

Model Implementations. One of the important models that adopt possible worlds semantics to capture uncertainty and incompleteness in attribute values is the C-tables model [Imielinski 1984]. C-tables are relational tables whose attributes are represented using variables, and each tuple is associated with a Boolean condition on the attribute variables. A tuple belongs to the database, if and only if its associated condition is satisfied. Many of the proposed models afterwards, e.g.,

[Abiteboul 1987, Dalvi 2004, Cheng 2007], treat tuples' uncertainty and attributes' uncertainty separately by introducing two basic types of uncertainty quantified with probability values. The first type, usually referred to as "membership uncertainty" [Dalvi 2004], treats tuples as probabilistic events capturing the belief that they belong to the database. Possible worlds are thus viewed as conjunctions of tuple events. The second uncertainty type, referred to as "value uncertainty" [Cheng 2007] represents attributes as probability distributions on continuous or discrete domains of possible values, e.g., modeling readings of sensing devices, or data entry errors in dirty databases.

2.2.2.3 Queries Semantics

In queries semantics, we need to consider two possible semantics. In the first, the query is applied to every possible world, and the result consists of all possible answers: called the possible answer sets. In the second, the query is also applied to all possible worlds, but the set of tuples are combined, and a single set of tuples is returned: this is called *possible answers semantics*. For a query, it is impractical to represent all possible answers but it is convenient to consider one answer at a time.

Definition. *Assume a query Q and a probabilistic database D. A tuple t is a possible answer to Q if there exists a possible world W / $W \in D$ such that $t \in Q(D)$. The possible answers are $Q_{poss} = \{t_1, ..., t_n\}$ where $t_1, ..., t_n$ are possible answers. The certain answers are $Q_{cert} = \{t_1, ..., t_k\}$ where $t_1, ..., t_k$ are all certain answers.* There are two approaches to query evaluation: intensional and extensional approach.

Intensional evaluation approach. In intensional query evaluation [Fuhr 1997, Sadri 1995] the probabilistic inference is performed over a propositional formula called lineage. *The lineage* of a possible output tuple is a propositional formula over the input tuples in the database, which says which input tuples must be present in order for the query to return that output. The intensional approach uses complex events by using the tuple names as atomic events. The calculation of probabilities does not depend on a specific plan. The intensional semantics on

probabilistic databases consists of a set of worlds and the content of each world is the output of Q on the database.

Extensional evaluation approach. The extensional approach [Fuhr 1997, Sadri 1995] evaluate queries by reusing standard relational techniques, operators and plans. An extensional operator is an extended relational operator (e.g., join, projection, selection,),to manipulate tuple probabilities. Each extensional operator makes some assumptions on the input tuples (e.g., that they are independent or that they are disjoint) and computes the corresponding probabilities. An extensional plan is a query plan where each operator is an extensional operator. A safe query plan [Dalvi 2004] is an extensional plan that, furthermore, is guaranteed to compute all output probabilities correctly and a query is safe if it admits a safe plan. Thus, if a query is safe, not only can we evaluate it efficiently, but we can actually push down the entire query evaluation in a relational database engine, and achieve real scalability. On the other hand, unsafe queries do not have any safe plans. However, in practice, we can always use an extensional plan to compute any unsafe query and obtain some approximate probabilities. Under some restrictions, the probabilities returned by any extensional plan are guaranteed upper bounds of the correct probabilities.
After studying queries' evaluation approaches we need also to study how to rank results.

Ranking queries on uncertain data and top-k processing. Few research works have addressed the problem of answering ranking queries in the presence of data uncertainty. Supporting ranking queries on uncertain data has been first proposed in [Soliman 2008]. That work introduced a framework to rank uncertain data based on the "marriage" of traditional top-k semantics and possible worlds semantics. Along the same lines, [Hua 2008, Cormode 2009]proposed similar query semantics and processing algorithms. The uncertainty model in all these works assumes that tuples have deterministic single-valued scores, and they are associated with membership probabilities. In [Hua 2008], computing probabilistic ranking queries with a given probability threshold is addressed. Given a threshold i, the objective is to report each tuple whose probability to appear in the top-k

answers is at least i. The given techniques are based on dynamic programming formulation under tuple level uncertainty. The work in [Soliman 2010] adopts the attribute level uncertainty model for formulating uncertain rank join queries, and studies the integration of rank join processing with probabilistic ranking.

Some recent works have addressed the problem of computing a consensus ranking from a space of possible worlds. The work [Li 2009] gives an approximate algorithm for computing a consensus ranking under the Kendall tau distance. The impact of tuple-level and attribute-level uncertainty on ranking queries has been modeled and addressed by current proposals from different perspectives. In most proposals, the two uncertainty types are handled in isolation by assuming that the underlying uncertainty type is either tuple-level (e.g., [Hua 2008]) or attribute-level (e.g., [Soliman 2009]). An important distinction among proposals that handle attribute-level uncertainty is their ability to support discrete and/or continuous domains of the uncertain attributes. For discrete uncertain attributes, a mapping can be constructed to model attribute-level uncertainty as tuple-level uncertainty, and hence leverage the ranking techniques developed for tuple-level uncertainty.

Formulating and processing top-k queries have received a considerable attention in the last years ([Ilyas 2008]is a thorough survey). Research works in the field adopt one of two ranking models: (i) top-k selection and (ii) top-k join. In the top-k selection model, scores are attached to base tuples, and the query reports the k tuples with the highest scores. The NRA (No Random Access) algorithm[Fagin 2003] is one of the prominent top-k techniques that adopt the top-k selection model. The input to the NRA algorithm is a set of sorted lists, each ranks the *same* set of objects based on one scoring predicate. The output is a ranked list of these objects ordered on the aggregate input scores. In the top-k join model, scores are assumed to be attached to join results rather than base tuples. A top-k join query joins a set of relations based on a given join condition, assigns scores to join results based on some scoring function, and reports the top-k join results. Many top-k join techniques address the interaction between computing the join results and producing the top-k answers. One example is the Rank-Join

algorithm [Ilyas 2004], which efficiently integrates the joining and ranking tasks. Integrating tuples' scores and probabilities as two interacting ranking dimensions is a recent issue [Soliman 2009]. Most current top-k processing proposals assume deterministic data, and hence they are not explicitly designed to treat probability as an additional ranking dimension.

2.3 Related work

2.3.1 Web services composition and ranking

Several mashup editors have been introduced by the industry with the objective of making the process of mashups creation as simple and "programmable-free" as possible. Examples include Yahoo Pipes [6], Google Mashup Editor [7], Intel Mash Maker [8]. These products allow average users to create mashups without any programming involved; the users need just to drag and drop services, operators and/or user inputs and to visually connect them. However, the knowledge required from users is not trivial because they are still expected to know exactly what the mashup inputs and outputs are, and to figure out all the intermediate steps needed to generate the desired outputs from the inputs. This includes selecting the needed services/data sources, mapping their inputs and outputs to each other and probably adding some mediation services/functions when inputs/outputs don't fit each other.

The Web Service Management System [Srivastava 2006] models data services as relations and allows users to mashup data services by expressing their mashup queries directly in terms of these relations. Along the same lines, the Web Service Mediator System WSMED [Sabesan 2012] allows users to mashup data services by defining relational views on top of them (called the WSMED Views). Unfortunately, users in these systems are assumed to have an understanding of the semantics of the data services that are available to them to be able to formulate their queries. Furthermore, users are supposed to import the services

[6] Yahoo Inc, Yahoo Pipes, http://pipes.yahoo.com/pipes/

[7] Google Inc. Google Mashup Editor, http://code.google.com/gme/

[8] Intel Inc. Intel Mash Maker. http://mashmaker.intel.com/web/

relevant to their needs; define views on top of imported services and enhance the views with primary-key constraints. These tasks are difficult and hinder average users from mashing up data services at large. These systems model data services as relations with inputs and outputs as the relations' attributes. This modeling is poor in semantics, as opposed to the use of domain ontologies, and may lead to ambiguity when data services have similar inputs and outputs (attributes), but different semantics. For example, assume a service $S(\$a, ?b)$ where a represents a school and b the student, the service S can return students or successful students in a given school. Compared to these works and to other academic mashup systems (e.g., [Tatemura 2008]), users of our system are not required to select the services manually, connect them to each other and drop code (in JavaScript) to mediate between incompatible inputs/outputs of involved services. This is completely carried out by the system in a transparent fashion. That is, our approach is *declarative*; users need just to specify the information they need without specifying how this information is obtained. Furthermore, the systems mentioned above do not provide any effective means to rank the data results returned by the mashups.

Our work is also related to the works around top-k queries and data ranking. Ranking queries are becoming dominant in many domain applications such as *multimedia databases*, *middelwares*, and *datamining*. The increasing of ranking works support ranking mainly in relational database management system and recently pay an attention of the research community. The previous research works [Ilyas 2008] in that area adopt one of two ranking models: *(i) top-k selection* and *(ii) top-k join*. In the top-k selection model, scores are attached to tuples in one single relation, and the query reports the k tuples with the highest scores [Fagin 2003]. Another research work attempted to integrate these two models [Ilyas 2004]. In contrast, the ranking in our system is complete: it computes the grades of individual data sets returned by data services and the integration thereof to compute the rankings of the final results based on the users fuzzy preferences. In addition, our approach adopts a flexible approach for preference modeling (i.e., a fuzzy approach) and provides two ranking models: scalar and vector models.

2.3.2 Uncertainty in web services composition

A considerable past work studying uncertain data management. In [Dong 2009], a Local-as-View-like data integration system was proposed for uncertain data sources. However this work has addressed only the issues of creating the probabilistic mappings between the mediated schema and the data sources' schemas, and queries transformation based on these probabilistic mappings. Along the same lines, the authors of [Magnani 2010] survey the different approaches (and the used formalisms) for the construction of probabilistic mappings, and the corresponding query transformation mechanisms. However, all of these works have addressed the uncertainty at the schema level only; i.e., the uncertainty resides in the way sources can be mapped to the mediated schema. In contrast, our project complements these works by addressing the uncertainty at the data level; i.e., we assume that there is a deterministic mapping between the sources' schemas and the mediated schema, and focus on computing the probabilities of the integrated data when the data inside the sources are themselves uncertain. Few research works have addressed the uncertainty at the data level. A query rewriting based approach was proposed in [Dalvi 2011] to speed up the query evaluation over uncertain data sources by exploiting the previously answered queries (which are stored as materialized views). In that work, authors define a partial representation for the materialized views. This representation describes whether the tuples inside the view are "independent" or "disjoint". However, this approach is limited because when the correlation between tuples is more complex than the independent and disjoint relationships the materialized views become useless. Also, it did not address the issue of finding the query plans that could give the right probabilities for the returned results. In [Agrawal 2010], the authors revisit the main data integration concepts (including the query containment, the certain answers, etc.) in the context of uncertain data sources. However, the issue of computing the tuples' probabilities and the impacts of the potential correlations were not studied. In [Hadjali 2008], an approach was proposed to match queries and views involving some fuzzy predicates. The approach returns the views that do provide answers whose satisfaction degree is over a threshold specified by the user. However, in that work, the tuples' uncertainty was not taken into account. In addition to all of

the cited limitations, none of these previous works have addressed the data ranking issue in the data integration context and the usage of probability as a new ranking dimension (in addition to that of the tuples' scores) in the ranking space, which is the main focus of this project.

The uncertainty of the data returned by data services is one of the key issues that have never been explored yet. Uncertainty is an inherent feature of the results returned by data services in many applications including Web data integration [Agrawal 2010, Soliman 2010], scientific data exploration [Buneman 2006], sensors networks [Tatbul 2004], objects tracking, etc.

Several works have focused on creating and modeling Data Web services [linh Truong 2009, Carey 2007]. In [linh Truong 2009], the authors proposed an XML-based modeling for data Web services along with a platform (called AquaLogic) for building data Web services on top of heterogeneous data sources. [Carey 2007] identified the different data quality aspects that a data Web service should specify in its description. Unfortunately, these works do not pay any attention to the uncertainty character that may be associated with the services' accessed data, nor provide effective means for an automatic selection and composition of data Web services.

A considerable body of works has addressed the services composition problem [Balbiani 2009, Wu 2009]. Most of these works are inspired by the Artificial Intelligence (AI) planning techniques; i.e., they are based on (1) transforming the WS composition problem into an AI planning problem and (2) on the use of AI planning techniques to automate the service composition. In [Wu 2009], the authors proposed a Bayesian-based approach to select the services compositions (called sequences) that has the largest probability as the best choice among the possible ones. In [Balbiani 2009], authors model Web services as automata executing actions and formalize the problem of computing Boolean formulas characterizing the conditions required for services to answer the client's request. Unfortunately, these composition approaches take into account only SaaS (Software-as-a-Service) Web services. They are inappropriate for the class of services we are targeting in our work, i.e., the Data Web services, which cannot be modeled as actions to apply the AI planning techniques [Barhamghi 2010]. Moreover, the uncertainty aspect was never looked at in these works.

2.3.3 Safe plans

Safe queries and safe query plans are introduced by [Dalvi 2004]. They also prove a dichotomy into polynomial time and #P-hard for conjunctive queries without self-joins over tuple independent tables. Olteanu et al[Olteanu 2009] address the problem of safe plans, by decoupling the data processing part of the query plan from the probabilistic inference part. They introduce a new type of plan that allows the optimizer to choose the best plan for the data processing part, yet allowing the probabilistic inference to take advantage of the query's safety. In that framework, a safe plan is an eager plan, where all probabilistic computations are performed as early as possible; lazy plans are at the other extreme as they compute the probabilities after the result tuples are computed.
[Gatterbauer 2010] introduced the dissociation technique; they define an order between query plans and thus approximate the query probabilities with best possible upper bounds in a database independent way. Query evaluation on BID tables was first discussed by Andritsos et al. [Andritsos 2006] and Re et al. [Re 2006]. Only conjunctive queries without selfjoins have been studied over BID tables.
[Sen 2007] discuss query evaluation over probabilistic databases represented by a graphical model. An optimization to query processing over such probabilistic databases is described by [Sen 2008]. The optimization finds common subgraphs in the GM and applies technique similar to lifted inference [Poole 2003], this can be very effective over large databases because they tend to have a large number of similar repeated subgraphs.
All approaches mentioned above cannot be applied in the case of uncertain data services. Thus, we proposed a set of conditions to check the safety for independent and BID tuples.

2.4 Conslusion

In this chapter, we presented the main concepts around Web service technology. We also introduced the reader to fuzzy sets, uncertainty in databases and the

probabilistic theory.
Based on the key concepts and the notions reviewed in this chapter, the next three chapters present our approach to compose uncertain data services.

Chapter 3

A Preference-Aware Query Model for Data Services

Contents

3.1 Introduction 31

3.1.1 Motivating Example 31

3.1.2 Challenges 32

3.1.3 Contributions 33

3.2 Preferences-Based Data Service Composition Model . . . 34

3.2.1 Preference Queries 34

3.2.2 Data Services 36

3.2.3 Query Rewriting 38

3.3 A ranking-aware algebra for data services compositions . 41

3.3.1 Scalar Grade based Results Ranking Algebra 42

3.3.2 Vector Grade based Results Ranking Algebra 46

3.3.3 Approach Overview 48

3.4 Conclusion 49

3.1 Introduction

Data services compositions (a.k.a., *data mashups*) are *situational applications* (i.e., applications that come together for solving some immediate business problems) that combine data elements from different data sources to provide value-added information for immediate business data needs. Typically, the access to these data sources is carried out through data Web services [Carey 2008, linh Truong 2009]. Data services composition has become so popular over the last few years; its applications vary from addressing transient business needs in modern enterprises [Guinard 2010, Jhingran 2006] to conducting scientific research in e-science communities [Zhao 2008].

Data services composition involves several challenging tasks including: selecting the data services that are relevant to user's needs, mapping their inputs and outputs to each other (and probably applying some mediation functions when inputs/outputs don't fit each other) within a composition plan. In addition, user preferences are an important factor that could be used to customize the composition. A more general and crucial approach to represent preferences is based on the fuzzy sets theory [Hadjali 2008, Dubois 2000]. Fuzzy sets are very appropriate for the interpretation of linguistic terms, which constitute a convenient way for users to express their preferences. For example, when expressing preferences about the price of an apartment, users often employ linguistic terms like cheap, affordable and not expensive.

3.1.1 Motivating Example

Consider a Web user *Melissa* planning to buy a new apartment. Melissa would like to find an apartment in a clean city, with an affordable price and located near to high schools with cheap tuition fees and good reputation. Melissa needs to retrieve cheap schools along with their tuitions fees and addresses from *nces.ed.gov* and their ratings from *psk*12.*com*. She needs then to connect to some e-commerce sites (e.g., *apartments.com*) to locate cheap apartments near to the schools found. She needs also to connect to the Outdoor City Pollution Database on *who.int* to filter out apartments located in polluted cities. Assume that these information are provided by the data services in Table 3.1. The service S_1 returns the schools, along

with their tuitions fees, reputations and addresses at a given country; S_2 returns the apartments for sale along with their prices at a given city, and S_3 returns the pollution level at a given city. Input and output parameters are proceeded by "$" and "?" respectively. Obviously, Melissa can answer her query by composing the following services.

Table 3.1: Available Web Services

Service	**Functionality**
$S_1(\$c, ?s, ?t, ?r, ?a)$	Returns the schools s along with their tuition fees t, reputation r and addresses a in a given country c
$S_2(\$a, ?ap, ?p)$	Returns the apartments for sale ap, their prices p at a given address a
$S_3(\$a, ?po)$	Returns the pollution level po for a given city a

3.1.2 Challenges

Mashing-up data services presents many challenges for the service composer (i.e., Melissa):

- ***Understanding the semantics of data services.*** Melissa needs to delve into the data service space and understand the semantics of each individual service in order to identify the services that may contribute to the resolution of her query. The semantics of a data Web service resides not in "how" inputs and outputs are related to each other but also in the fuzzy constraints; i.e., many services may have the same functionality types, but have completely different constraints. In the lack of a clear semantics definition inside the service description files, service composers will miss much of the services that are relevant to their queries; even worse they may wrongly select services that are irrelevant to their needs.

- ***Selecting and composing relevant data services.*** Let us assume now that Melissa is able to understand the semantics of available data services,

the next step would be to select participant services which better satisfy the user's preferences, figure out their execution order, and generate the orchestration composition plan that better answer a fuzzy preferences query.

- ***Ranking the results and selecting the best ones.*** Let us assume now that Melissa was able to create and execute the composition. Each of the composed services may return a huge number of business objects (e.g., apartments, schools) that may more or less match the user's preferences. As a result, Melissa will be overwhelmed with a great number of answers and may miss the ones that are most relevant to her needs.

3.1.3 Contributions

In this chapter we propose a declarative approach for composing data web services on the fly. Based on a semantic model for data web services, and a "declarative" composition query formulated against domain ontologies along with a set of preferences formulated as fuzzy constraints, our proposed composition system generates detailed descriptions of the composition that fulfills the query. The generated composition plan ranks also the results at the execution time. We summarize below our major contributions:

- ***A semantic model for data services.*** We propose to model data services as *RDF Views* over domain ontologies. An RDF view allows capturing the semantics of the associated service in a "declarative" way based on concepts and relations whose semantics are formally defined in domain ontologies. The semantic model allows characterizing the preferences as fuzzy constraints using the fuzzy set theory.

- ***A declarative model for composing data services.*** We propose to use the mature query rewriting techniques for composing data services. First, we select the relevant services which satisfy fuzzy users' preferences and rewrites the query in terms of calls to selected services. Our composition model enables average users to compose data services as all what they need to do is just specifying their data needs declaratively.

- ***A data ranking model to select the best answers.*** We propose a ranking-aware composition algebra (and implementation thereof) that allows ranking the returned results based on the user preferences at the composition execution time.

The remainder of this chapter is organized as follows. In Section 2, we present our declarative approach to construct service compositions. We show also our semantic modeling of data services and queries; present the composition algorithm through an example. In section 3 we introduce our ranking-aware composition algebra and we give an overview of our approach. Finally, in section 4, we summarize our contributions and conclude.

3.2 Preferences-Based Data Service Composition Model

In this section we present a 'declarative approach for data services composition that addresses the challenges discussed in the previous section. We show the different phases involved in data service composition, starting from the service modeling to the generation of the final composition that will be returned to users.

3.2.1 Preference Queries

We adopt a declarative approach to Web services composition, i.e., instead of selecting and composing Web services manually, users formulate their composition queries over domain ontologies. We consider conjunctive preference queries expressed over domain ontologies using a slightly modified version of SPARQL, the de facto query language for the Semantic Web.

Users express their preference queries over domain ontologies using a slight modification of SPARQL. Figure 3.1 gives the formulated query for the running example in (c), and its graphical representation in (a). The user's preferences are expressed in the "PREFERRING" clause. We model user's preferences using the fuzzy sets theory [Zadeh 1965].

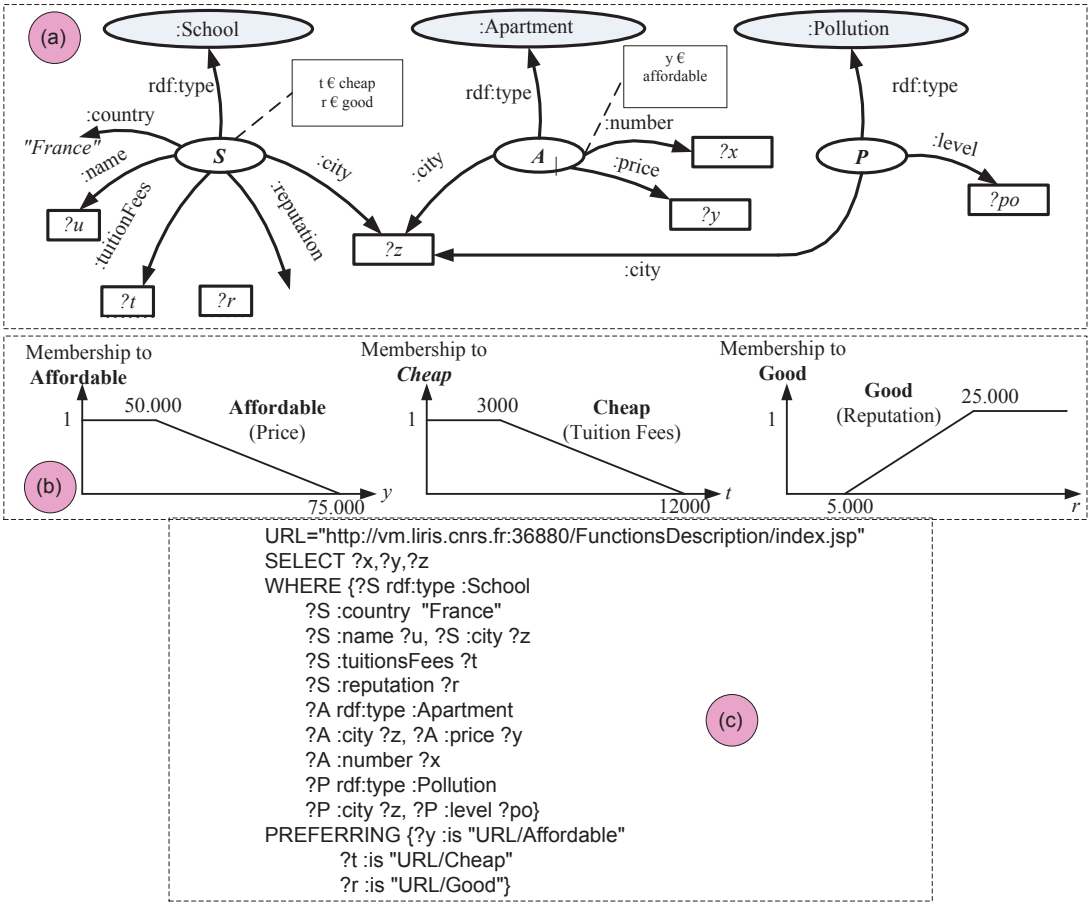


Figure 3.1: (a) a Graphical Representation of the Query, (b) the Associated Membership Functions, (c) the formulated query in SPARQL

Formally, preferences are a set of constraints C_i of the form : x *is* $FuzzyTerm$, where x is a variable, and the $FuzzyTerm$ is a fuzzy term (e.g., "Cheap") interpreted according to a membership function $\mu_F : X \rightarrow [0, 1]$, specifying for each value of x the grade (i.e., $\mu_F(x)$) to which x belongs to $FuzzyTerm$. Note that $\mu_F(x) = 1$ reflects full membership of x to $FuzzyTerm$ and $\mu_F(x) = 0$ absolute non-membership. For example, the fuzzy terms used in Figure 3.1 (part-b) (i.e., *Affordable* price, *Cheap* tuition fees, and *Good* reputations) are interpreted using the membership functions shown in Figure 3.1 (part-b).

Membership functions of fuzzy terms are implemented as Web services and

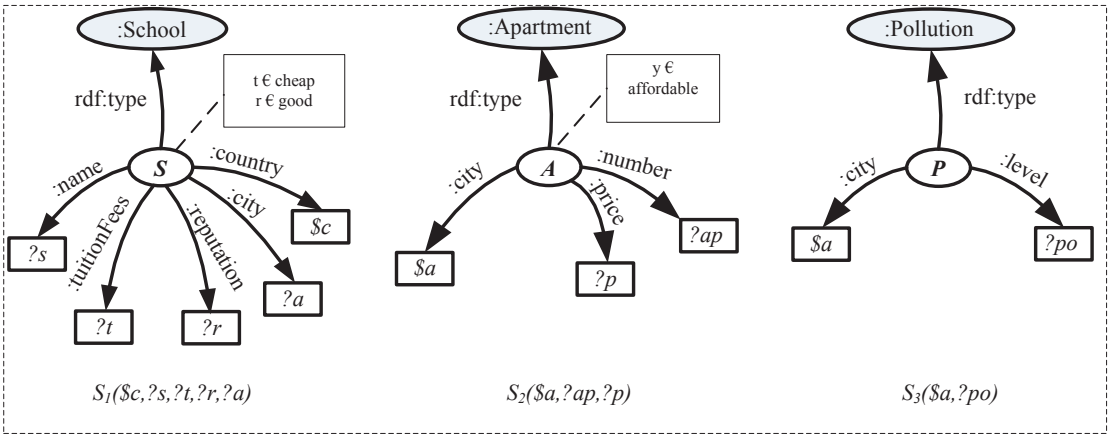


Figure 3.2: RDF Parametrized Views - Examples.

can be shared by users. They are used in the PREFERRING clause of the query where the URL of the implementing Web service is mentioned. The head and body of a query Q are defined in SELECT and WHERE clauses respectively. For instance, $?y$ is 'URL/affordable' means that the user prefers services that provide apartments with affordable price. The semantics of affordable is given in $URL = "http://vm.liris.cnrs.fr:36880/FunctionsDescription/index.jsp"$.

3.2.2 Data Services

The functionalities of data services, as opposed to traditional Web services that encapsulate software artifacts, can be only captured by representing the semantic relationship between inputs and outputs [Yu 2008, Barhamghi 2010]. Therefore, we model data services as RDF Parametrized Views (RPVs) over domain ontologies Ω. Each view captures the semantic relationships between input and output sets of a data service using concepts and relations whose semantics are formally defined in ontologies. Formally, a data service S_i is described over Ω as a predicate $S_i(\$X_i, ?Y_i) :- < F(X_i, Y_i, Z_i), C_i >$, where:

- X_i and Y_i are the sets of input and output variables of S_i, respectively. Input and output variables are also called as distinguished variables. They are prefixed with the symbols "$"and "?" respectively.

- $F(X_i, Y_i, Z_i)$ is the functionality of the service and represents the semantic relationship between input and output variables. Z_i is the set of existential variables relating X_i and Y_i. $F(X_i, Y_i, Z_i)$ has the form of RDF triples where each triple is of the form *subject.property.object*.

- C_i is a set of value constraints expressed over the X_i, Y_i or Z_i variables. C_i may include fuzzy constraints to characterize the data manipulated by S_i.

Each data service requires a particular set of inputs (parameter values) to retrieve a particular set of outputs; i.e., outputs cannot be retrieved unless inputs are bound. For example, one cannot invoke data service S_2 without specifying the address for which it need to know the apartment for sale and the price. Inputs and Outputs are prefixed with '$' and '?', respectively in the head of the view $S_i(\$X_i, ?Y_i)$. X_i and Y_i variables are defined in the WSDL description of data services.

The data services S_1, S_2 and S_3 are described by the following RDF view where "*School*", "*Apartment* and "*Pollution* are the ontological concepts, "*country*", "*name*", "*tuitionfees*", "*reputation*", "*city*", "*number*", "*price*" and "*level*" are the different attributes.

```
S1($c,?s,?t,?r,?a):-
               F:{ ?S rdf:type :School,
                   ?S :country       $c,
                   ?S :name          ?s,
                   ?S :tuitionFees   ?t,
  ?S :reputation ?r,
  ?S :city ?a}

               Constraints: {?t is URL/CHEAP, ?r is URL/GOOD}

S2($a,?ap,?p):-
               F:{ ?A rdf:type :Apartment,
                   ?A :city    $a,
                   ?A :number  ?ap,
```

```
        ?A :price   ?p}
    Constraints: {?p is URL/AFFORDABLE}


S3($a,?po):-
    F:{ ?P rdf:type :Pollution,
        ?P :city    $a,
        ?P :level   ?po}
```

Figure 3.2 gives graphical representations of the RDF Parametrized Views of our sample services. Note that, the current Web service description standards (e.g., WSDL) can be extended straightforwardly with our proposed modeling to data services, as RPVs can be incorporated within the description files (e.g., WSDL) as annotations.

3.2.3 Query Rewriting

In a previous work [Barhamghi 2010] we proposed an efficient RDF query rewriting algorithm. We exploit that algorithm to compose data services. Given a composition query Q and a set of data services represented by their corresponding RPVs $V = v_1, v_2, ..., v_i$, the algorithm rewrites Q as a composition of data services whose union of RDF graphs (denoted to by G_V) covers the RDF graph of Q (denoted to by G_Q). The rewriting algorithm has two phases:

3.2.3.1 Phase-I: Finding Relevant Sub-Graphs

In the first phase, our composition system compares G_Q to every RPV v_i in V and determines the *class-nodes* (i.e., the variables in Q whose types are ontological classes, e.g., “A", “S" and “P" in Figure 3.1) and object properties in G_Q that are covered by v_i. The system stores information about covered class nodes and object properties as a partial containment mapping in a *mapping table*. The mapping table points out the different possibilities of using an RPV to cover parts of G_Q. In this phase, the rewriting algorithm considers each class-node and each object-property in Q and tries to determine the rel-

evant views to them. A view is said to be relevant in one of the following two cases:

Case1: Covering Class-nodes. v_i includes a class-node C_v whose type C is the same as the type of a class-node C_Q in Q such that the following conditions hold true:

- If the mapped class-node C_Q has a distinguished variable x in the query , i.e. a datatype property of C_Q is bound to a distinguished variable in Q , then either the same datatype property of C_v is projected in v_i (i.e. it is bound to a distinguished variable in v_i), or it can be recovered because all datatype properties used in the skolem function of C are projected in v_i and thus can be used to recover the missing distinguished variable (i.e. the missing datatype property of C_v) of C_v.

- If the mapped class-node C_Q has an existential variable x in the query Q (i.e. one of its datatype properties binds to an existential variable x in Q).

- If the mapped class-node C_Q has a constant in its triples group in the query, then either the view has to project the datatype property of C_v that corresponds to the constant, or such datatype property can be recovered.

- If the mapped class-node C_Q is involved in an object-property in the query, then the view has either to project the attributes of the skolem function of so that to enforce the join with object-property or it has to cover the object-property as defined by the Case2.

Case 2 (Covering Object-properties). v_i includes an object-property p of Q in its definition such that the class-nodes linked by p can be mapped to the corresponding class-nodes of p in Q(i.e. they have the same types). The view v_i is relevant to the query if it projects the datatype properties used in the skolem function of each of the class-nodes linked by , or it covers the class-nodes for which it does not project the datatype properties used in their skolem functions.

Example. The service S_1 has a class node $S_1.S$ that can be matched with $Q.S$. All the data-type properties of $Q.S$ that bound to distinguished variables in

Service	Covered classnodes & properties
$S_1(\$c, ?u, ?t, ?r, ?z)$	$Q.S(\text{“}france\text{”}, u, t, r, z)$
$S_2(\$z, ?x, ?y)$	$Q.A(z, x, y)$
$S_3(\$z, ?po)$	$Q.P(z, po)$

Table 3.2: Mapping Table: the covered sub graphs by sample data services

Q also bound to distinguished variables in S_1. Furthermore, $Q.S$ is involved in a join over the variable $?z$ with the class-nodes $Q.A$ and $Q.P$. Even though S_1 does not cover the class-nodes $Q.A$ and $Q.P$, the join over $?z$ can be still enforced as $?z$ is a distinguished variable in S_1. Therefore, S_1 can be used to cover $Q.S$, and thus inserted in the Table 3.2. The same discussion applies to S_2 and S_3.

3.2.3.2 Phase-II: Generating data service compositions

In the second phase, the RDF query rewriting algorithm explores the different combinations from the Mapping-and-Connectivity table. The algorithm needs to consider the combination of disjoint sets of covered object-properties and class-nodes except when some datatype properties are missing in a covered class-node, in which case additional class-nodes are added to recover missing datatypes properties. We consider disjoint sets of covered object-properties and class-nodes for the following reason: each line in the mapping table contains a class-node CN_i or an object-property OP_i along with the minimum set of classnodes/object-properties (CNs/OPs) that are linked with that class-node/object-property (CN_i/OP_i) via some joins that cannot be enforced if other class-nodes/object-properties (CNs/OPs) from a different view were used in the combination (this happens when the joins are made over existential variables in the view). This assumption speeds up the second phase of the rewriting algorithm because it prunes the combinations with joins that cannot be enforced.
A combination is a valid rewriting of (a valid composition) if the following two conditions hold true:
1- It covers the whole set of class-nodes and object-properties in Q, and

2- The combination is *executable*. A composition is said to be executable if all input parameters necessary for the invocation of its component services are bound or can be made bound by the invocation of primitive services whose input parameters are bound.

Example. Continuing with the running example, there is only one possible combination $C_1 = \{S_1, S_2, S_3\}$. Only $S_1(\$c, ?u, ?t, ?r, ?z)$ can be invoked at the beginning as its input parameter is bound. After the invocation of S_1, the variable z become available; hence, the services S_2, S_3 become invokable. Consequently C_1 is executable and is considered as a valid composition.

3.3 A ranking-aware algebra for data services compositions

The obtained composition for a query will be deployed as new permanent data Web service. For this purpose, two essential issues are addressed. First, similarly to traditional Web services composition, a composite Web service (i.e. a composition)needs to be translated into an execution plan describing both data flow and intermediary data processing among individual web services in a composition. Each service occurrence in the composition (that is obtained from the query rewriting algorithm) will be translated to an "invoke" operation. The outputs of similar web services (services covering the same portion in the query) will be unified by a "'union"' operation that is responsible for removing redundant tuples. "'Join"' operations will be used to feed a service with data tuples coming from its parents in the composition, it joins tuples from parent services in a composition. "'Select"' operations are used to filter out tuples that do not satisfy a specified equality or order constraint. Second, an efficient execution engine that is capable to understand and execute the plan's building constructs needs to be implemented. Component services must be executed in a particular order depending on their access patterns. If a service S_i has an input x that is obtained from an output y of S_j then S_i must be preceded by S_j in the execution plan

Table 3.3: Implemented norms and conorms

Name	$TNorm : \top(x, y)$
Zadeh	min(x,y)
Probabilistic	xy
Lukasiewicz	$max(x + y - 1, 0)$
Hamacher	$\frac{xy}{\gamma+(1-\gamma)(x+y-xy)}$
Weber	$\begin{cases} x & \text{if} \quad y = 1 \\ y & \text{if} \quad x = 1 \\ 0 & \text{else} \end{cases}$

Name	$Conorm : \perp (x, y)$
Zadeh	$max(x, y)$
Probabilistic	$min(x + y, 1)$
Lukasiewicz	$max(x + y - 1, 0)$
Weber	$\begin{cases} x & \text{if} \quad y = 0 \\ y & \text{if} \quad x = 0 \\ 1 & \text{else} \end{cases}$

In this section, we propose an algebra to orchestrate the data services selected in the previous steps. The proposed algebra allows ranking the returned results based on their relevances to user's preferences. Results ranking is important as the results number may be very large which may cause the users to miss the ones that are most relevant to their needs. To enable ranking-aware query processing, our proposed algebra relies on the mature fuzzy database foundations [Dubois 1999]. This new algebra enables and determines our query execution model and operator implementations.

We describe below two sets of ranking-aware composition operators that follow two approaches to rank data: (i) scalar grades based ranking, and (ii) vector grades based ranking.

3.3.1 Scalar Grade based Results Ranking Algebra

The operators in this set assume that each manipulated tuple is associated with a grade computed as the aggregation of the different grades associated to its attributes that are involved in fuzzy preferences. We define the following operators:

- *The Grade-aware Invocation* $Invoke^g(S, t^g_{in}, O^g)$: Let S be a service, t^g_{in} the graded input tuple with which S is invoked, O^g the graded output, and $S.O$ be the output returned by S. The $Invoke^g$ operator relays the tuples from $S.O$ to O^g, and for each relayed tuple t_i it computes the grade $g(t_i)$ as follows. First, assume t_i is involved in n preference fuzzy constraints P_j(where $1 \leq j \leq$n), the operator computes $g_1(t_i) = \top(\mu_{P_1(t_i)}, \mu_{P_2(t_i)}, ..., \mu_{P_n(t_i)})$ where $\top$ is a t-norm operator (that generalizes the conjunction operation) and μ_{P_i} the membership function associated with P_i. We implemented the T-norms presented in Table 3.3. The Zadeh t-norm is the greatest t-norm, thus leading to an optimistic aggregation strategy. The Lukasiewicz and Weber t-norm yield a pessimistic aggregation strategy. Second, it computes $g(t_i)$ as follows: $g(t_i) = \top(g(t_{in}), g_1(t_i))$.

- *Graded Join*: $\infty^g(I^g_1, I^g_2)$, where I^g_1 and I^g_2 are two graded data sets. The grade of an outputted tuple is given by:
$g(\infty^g(t, t')) = \top(g(t), g(t'))$ where $\top$ is a t-norm, and t and t' are tuples from I^g_1 and I^g_2 respectively.

- *Graded Projection* $\prod^g_A$. The projection is an operation that selects specified attributes $A = \{a_1, a_2, ...\}$ from a results set. The grade of an outputted tuple t is: $g(t) = \perp (g(t'_1), .., g(t'_i), .., g(t'_n))$ where $t = \prod_A(t'_i)_{i=1:n}$ and $\perp$ is the co-norm corresponding to the t-norm $\top$ used in the graded join.

- *Graded Union* $\cup^g$. The grade of an outputted tuple t is:
$g(t) = \perp (g(t'_1), .., g(t'_i), .., g(t'_n))$, where $t'_i = t$ and $i = 1 : n$

- *Graded Rank* $Rank^g$: the rank operator orders all outputted tuples according to assigned grades. Let t_1, t_2 be two tuples and g_1, g_2 be the grades respectively. If $g_2 \leq g_1$ so t_1 appears before t_2.

- *Graded Select* $Select^g$: Let $\overline{c}$ be a set of conditions; The probability of a tuple t in the outputted set is computed as follows:

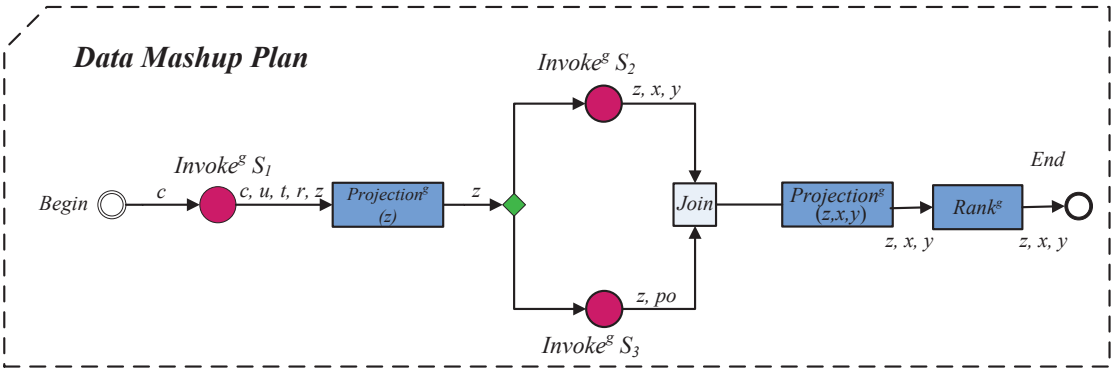


Figure 3.3: Composition Plan

$$prob(t) = \begin{cases} prob(t) & \text{if} \quad \overline{c} = true \\ 0 & \text{if} \quad \overline{c} = false \end{cases}$$

Example. We explain the previous operators based on our motivating example. The services S_1, S_2 and S_3 can be composed to find the apartments for sale located in cities with low pollution levels and near to schools with good reputations in a given country as shown in Figure 3.3. First, S_1 is invoked with the desired country (e.g., France). The invocation operator $Invoke^g(S_1)$ computes the grades of obtained tuples. The $Projection^g(z)$ operator projects the obtained tuples on the city attribute (i.e., z). Then, the obtained cities are used to invoke S_2 to retrieve nearby apartments along with their prices. In parallel, S_3 is invoked to retrieve the pollution levels of obtained cities. The results of S_2 and S_3 are joined over the variable z. Then, the $Projection^g(z, x, y)$ operator retains only the apartments information (i.e., numbers, prices and cities). All of these operators compute the tuples' rankings according to our defined equations presented earlier. Figure 3.4 shows the results (along with their rankings) at the output of each of these operators, and the final results at the composition's output.

- The $Invoke^g(S_1)$ operator invokes S_1 with the value "France", and computes

the different grades. For instance the grades of the school "s3" are computed as follows: based on the membership functions associated with the tuition fees and the reputation fuzzy predicates, the grade of the *tuition fees* attribute is 0.22 and the grade of the *reputation* attribute is 0.05. Hence, $Grade_{zadeh}(s3) = min(0.22, 0.05) = 0.05$, $Grade_{probabilistic}(s3) = 0.22 * 0.05 = 0.011$, and $Grade_{Lukasiewicz}(s3) = max(0.22 + 0.05 - 1, 0) = 0$.

- The $Projection^g(z)$ operator projects the obtained tuples on the city attribute (i.e., z) and computes the grades of obtained tuples. For example, the grade of the outputted tuple corresponding to "Lyon" is computed as follows:
  $Grade_{zadeh}(Lyon) = max(1, 0.4) = 1$
  $Grade_{probabilistic}(Lyon) = min(1 + 0.311, 1) = 1$
  $Grade_{Lukasiewicz}(Lyon) = max(1 + 0.178 - 1, 0) = 0.178$

- The $Invoke^g(S_2)$ operator invokes, for each input tuple, the service S_2 and computes the grades of obtained tuples. For example, the grade of the apartment "$a1$" (at the output of $Invoke^g(S_2)$) is computed as follows: given that the apartment "$a1$" accessed by S_2 has a grade of 1, and that the grades of the city "Lyon" at the input of $Invoke^g(S_2)$ are shown above, then the grades of "$a1$" at the output of $Invoke^g(S_2)$ are: $Grade_{zadeh}(a1) = min(1, 1) = 1$, $Grade_{probabilistic}(a1) = 1 * 1 = 1$, and $Grade_{Lukasiewicz}(a1) = max(1 + 0.178 - 1, 0) = 0.178$. The *join* operator joins S_2 and S_3 outputted tuples and computes the associated grades. For example, the grade of the tuple corresponding to "$a3$" is computed as follows:
  $Grade_{zadeh}(a3) = min(0.6, 0.52) = 0.52$
  $Grade_{Probabilistic}(a3) = 0.312 * 0.36 = 0.112$
  $Grade_{Lukasiewicz}(a3) = max(0.12 + 0.15 - 1, 0) = 0$

- Finally, the $Rank^g$ orders results in ascending order(from the most satisfactory to the least satisfatory).

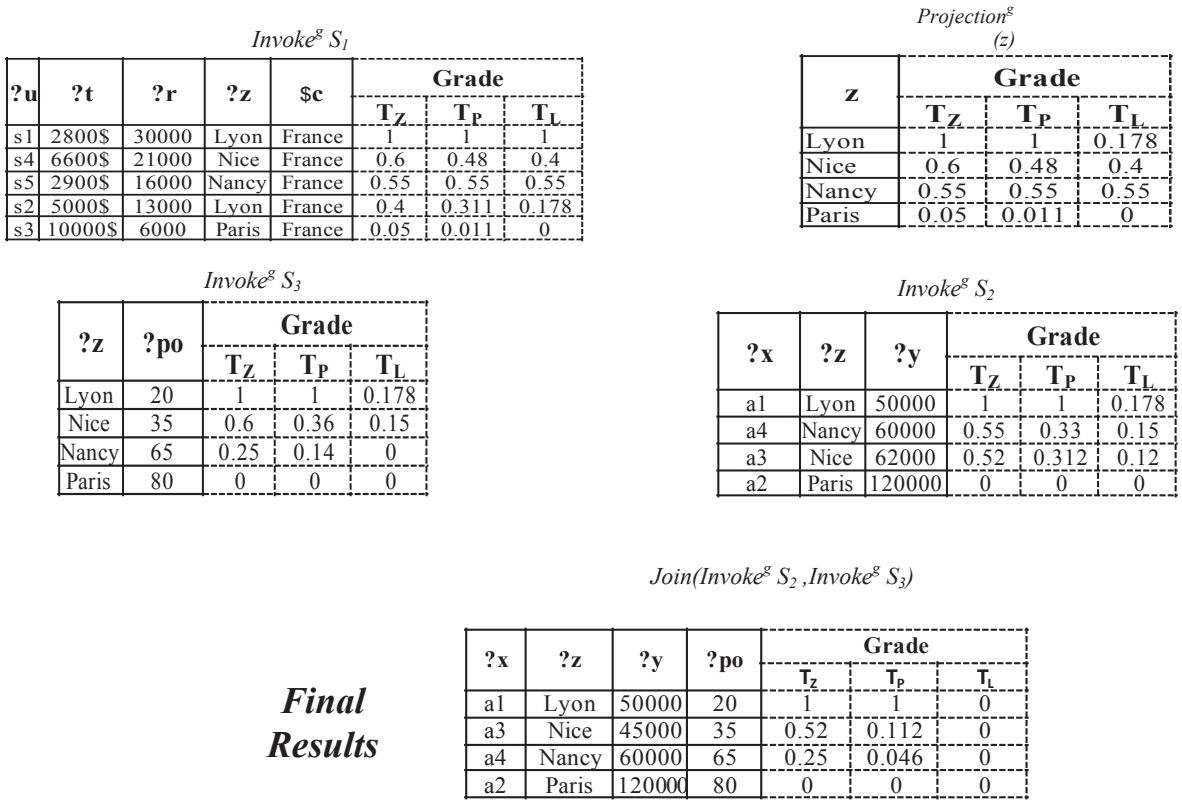


Figure 3.4: The intermediate and final results along with their grades

3.3.2 Vector Grade based Results Ranking Algebra

Merging different grades in one aggregated scalar grade is interesting but presents two main drawbacks. First, it does not allow users to know why a given tuple is a good or a bad result. Details on how fuzzy user preferences match data are not kept. Second, the tuples ordering may vary from one t-norm to another. To overcome these drawbacks, we propose to associate to each tuple a vector of grades. One may not always prefer to aggregate the different computed partial grades. In this case, each tuple t is associated with a vector of grades (instead of a scalar grade). To rank query results, one should revisit the above graded algebraic operators. For instance, The following set of revised graded operators are defined.

- *Graded Join* $\infty^g(I_1^g, I_2^g)$, where I_1^g and I_2^g are two graded data sets. The

revised grade of an outputted tuple is given by:
$g(\infty^g(t,t')) = (\top(g_1(t),g_1(t')),\cdots,\top(g_d(t),g_d(t')))$
where $\top$ is a t-norm and t (resp. t') is a tuple of the set I_1^g (resp. I_2^g), and $g_j(t)$ is the grade of the tuple t relative to a fuzzy predicate P_j.

- *Graded Projection* $\prod_A^g$. The grade of an outputted tuple t is: $g(t)= \{\perp (g_1(t'_1),..,g_1(t'_n)),..,\perp (g_j(t'_1),..,g_j(t'_n)),..,\perp (g_m(t'_1),..,g_m(t'_n))\}$ where $t = \prod_A(t'_i)_{i=1:n}$ and $\perp$ is the co-norm corresponding to the t-norm $\top$ used in the graded join, and $g_j(t')$ is the grade of the tuple t' relative to a fuzzy predicate P_j. The implemented co-norm are presented in Table 3.3.

- *Graded Union* $\cup^g$. The grade of an outputted tuple t is:
$g(t)= \{\perp (g_1(t'_1),..,g_1(t'_n)),..,\perp (g_j(t'_1),..,g_j(t'_n)),..,\perp (g_m(t'_1),..,g_m(t'_n))\}$
where $t'_i = t$, i=1:n and $g_j(t')$ is the grade of the tuple t' relative to a fuzzy predicate P_j.

- *Graded Select* σ^g. The grade of an outputted tuple t is:
$g(t)= \{(g_1(t'_1),..,g_1(t'_n)),..,(g_j(t'_1),..,g_j(t'_n)),..,(g_m(t'_1),..,g_m(t'_n))\}$ where $t'_i = t$, i=1:n and $g_j(t')$ is the grade of the tuple t' relative to a fuzzy predicate P_j.

Table 3.4 shows the final answers along with the different grades (for the fuzzy constraints *Cheap*, *Good*, *Affordable* and *Low*).

Table 3.4: Vector Ranking results

ap	a	p	po	Cheap	Good	Affordable	Low
a1	Lyon	50000$	20	1.00	1.00	1.00	1.00
a2	Paris	120000$	80	0.22	0.05	0.00	0.00
a3	Nice	45000$	35	0.60	0.80	0.52	0.75
a4	Nancy	60000$	65	1.00	0.55	0.60	0.25

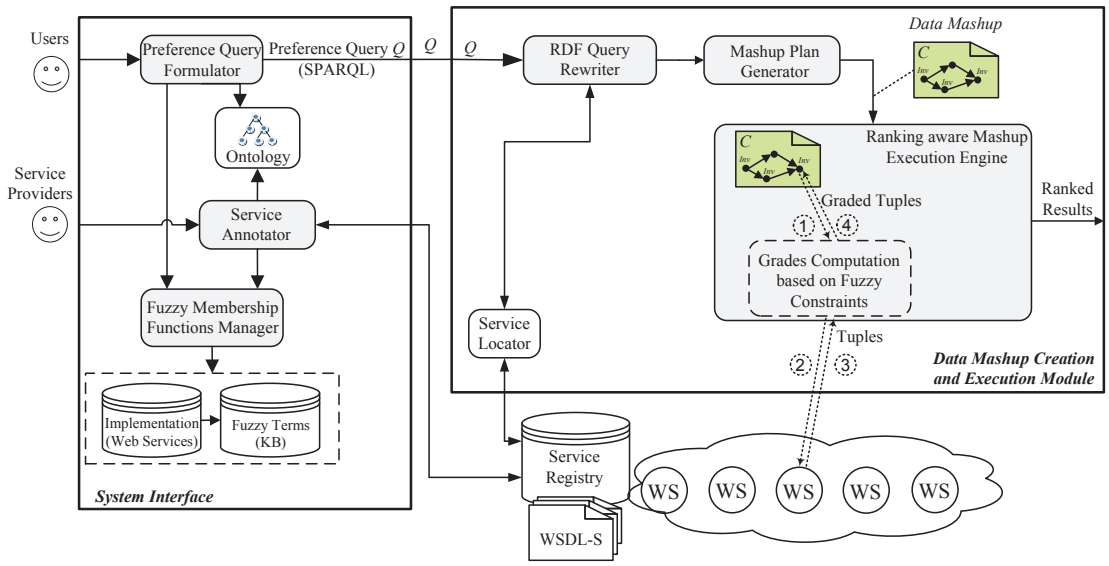


Figure 3.5: An overview of the proposed approach

3.3.3 Approach Overview

Figure 3.5 gives an overview of our proposed composition approach. Our approach is "declarative"; i.e., composition creators are relieved from having to select services and build manually the composition plan, a task that would generally require important programming skills. They need only to formulate their declarative queries over domain ontologies using the do facto ontology query language SPARQL[1].

The *Preference Query Formulator* component provides users with a GUI implemented with Java Swing to interactively formulate their queries over a domain ontology. Users are not required to have knowledge about SPARQL (or any specific ontology query languages) to express their queries, they are assisted interactively in formulating their queries and specifying the desired fuzzy terms.

The*Fuzzy Membership Functions Manager* component is used to manage fuzzy linguistic terms. It enables users and service providers to define their desired fuzzy terms along with the associated fuzzy membership functions. The defined terms are stored in a local fuzzy terms knowledge base which can be shared by users,

[1]http://sparql.org/

and are linked to their implementing Web services. Examples of fuzzy terms along with their services can be found on http://vm.liris.cnrs.fr:36880/FuzzyTerms. Users and service providers can directly test the proposed membership functions on that link and use the associated fuzzy terms. For each fuzzy term we provide a shape that gives a graphical representation of the associated membership function, a form that helps users to compute the degree to which a given value is in the fuzzy set of the considered fuzzy term, and a WSDL description of the Web service that implements the membership function.

RDF Query Rewriter implements an RDF query rewriting algorithm [Barhamghi 2010] to identify the relevant data services that match (some parts of) a user query. For that purpose, it exploits the annotations that were added to the service description files (e.g., WSDls). The Service Locator feeds the Query Rewriter with data services that most likely match a given query. Our approach exploits the mature query rewriting techniques [Barhamghi 2010] to fully automate the composition process.

The Service Annotator component annotates the service description files (e.g., WSDL files, SA-Rest, etc).
The composition plan generator orchestrates the selected services using a ranking-aware composition algebra that we have devised for that purpose. The composition will be then displayed to users, who will be able to execute it with their inputs.

The Ranking Aware Composition Execution Engine allows to execute our defined algebra. The execution engine assigns *grades* to results returned from services' calls based on their matching to users' preferences. We detail all of the previous steps in the subsequent subsections.

3.4 Conclusion

In this chapter, we proposed a declarative approach to compose data Web services on the fly. We proposed to model data services as RDF Views over domain ontolo-

gies to represent their semantics declaratively. Our semantic model allows characterizing the returned data using the fuzzy set theory. Our approach is based on the usage of the query rewriting techniques to automate the composition process, and allows to rank-order the composition results based on the user preferences, which are modeled as fuzzy constraints.

Chapter 4

Handling Uncertainty in Web Services Composition

Contents

4.1 Introduction 52

4.1.1 Motivating Scenario 52

4.1.2 Challenges 54

4.1.3 Contributions 55

4.2 Uncertain Data Services: A Probabilistic Model 56

4.2.1 A description model for uncertain data services 56

4.2.2 An Invocation Model For Uncertain Data Services 58

4.3 Uncertain Data Service Composition Model 59

4.3.1 Composition Semantics 60

4.3.2 An Algebra for Uncertain Data Services Composition 63

4.4 Block Independent Disjoint (BID) services 64

4.4.1 BID-P-Service Definition 65

4.4.2 BID-P-Service invocation 66

4.4.3 BID-P-Service composition 67

4.5 Conclusion 68

4.1 Introduction

Data services and Web services in general have received a considerable attention in recent years [Yu 2008]. Previous works have addressed the different aspects of the Web service life-cycle, including service creation, selection, discovery, invocation and composition [Yu 2008]. However, there are still many issues related to the quality of data services themselves that need to be explored [Dustdar 2012]. The uncertainty of the data returned by data services and their compositions is one of the key issues that have received little or no consideration, and which is the focus of this chapter.

4.1.1 Motivating Scenario

The Table 4.1 below gives examples of uncertain data services from the eCommerce domain. The service S_1 returns the information of a given product; S_2 returns the products which have been ordered by a given customer; S_3 returns the customers at a given city; S_4 returns the sales representatives along with their phone numbers at a given city. The uncertainty of data services could have different origins. A data service may be uncertain because it integrates different data sources adopting different conventions for naming the same objects set. For example, S_1 provides complete information about products by integrating two Web data sources *cdiscount.com*, and *amazon.com*. S_1 joins products from these two sources over the product name. However, the name of the same product may be stored differently in the two sources, e.g., *computer* in *cdiscount.com* versus *laptop* in *amazon.com*. Imprecise matches (usually quantified as numbers between 0 and 1) between products in the two sources can be interpreted as the probability the two products match; i.e. S_1 will return for each tuple obtained from the matching, the probability that tuple exists. This type of uncertainty is very common in the Web data integration domain [Agrawal 2010, Soliman 2010]. There is a rich literature on record linkage (also known as de-duplication, or merge-purge) [Marian 2011], that offers an excellent collection of techniques for computing matches or probabilities.

The uncertainty associated with S_2, S_3 and S_4 could come from the fact that the data sources accessed (or integrated) by these services contain conflicting information about customers and sales representatives (e.g., conflicting addresses for the same custumer, different phone numbers for the same representative, etc.). This is especially common in applications like sensor networks [Tatbul 2004] and object tracking [Cheng 2004], where the same data source may be fed by different sensors, that often report conflicting detections indicating several simultaneous locations for the same object. A common approach for storing such sensor data is to produce one record for each of the possible object locations, and assign a confidence degree to each record. Other common uncertainty sources may include privacy concerns [Fung 2012], where data items are deliberately made imprecise (e.g., the salaries are anonymized), or left out altogether; imprecise data production or collection methods as in the scientific data exploration domain [Buneman 2006] (e.g., imprecise scientific experiments, unreliable instruments, etc.).

Table 4.1: Examples of Data Web Services

Service	**Semantics**	**Service Type**
$S_1(\$p, ?pr, ?sh, ?cl)$	Returns informations (price pr, shape sh, color cl) about a given product p	Uncertain
$S_2(\$c, ?p, ?pr)$	Returns the products p along their prices pr which have been ordered by a given customer c.	Uncertain
$S_3(\$a, ?c, ?j)$	Returns the customers c and their jobs j at a given city a.	Uncertain
$S_4(\$a, ?s, ?t)$	Returns sales representatives s along with their phone t numbers at a given city a.	Uncertain

The uncertainty associated with uncertain services must be explicitly modeled and described in order to ensure that service consumers can understand and in-

terpret correctly the data returned by services and use them in the right way. For example, the consumer of S_2 should be advised about the probability of each returned tuple so that he can make the right product choice. The need for a clear uncertainty model for uncertain services is further exacerbated when they are composed to provide value-added services. For example, S_2 and S_3 can be composed to find the most ordered products in a given city. First, S_3 is invoked with the desired city (e.g., Lyon). Then, the obtained customers are used to invoke S_2 to obtain the ordered products. S_1 can be also included to retrieve the products' prices. If we neglect the probability metadata of the composition's outputs, we risk to select the products that appear first in the output list, and which may not be the most probable products. The importance of considering the uncertainty of output data becomes clearer when the output includes a sheer number of products where the most probable ones may not appear first, leading users to miss the most interesting results among the complete results list. The uncertainty of each of these services needs to be explicitly defined to be able to aggregate their returned data pieces and compute the probabilities of the composition's outputted results (which could be used for ranking these results).

4.1.2 Challenges

Handling the uncertainty associated with data services may involves several challenges:

- *Uncertain data services modeling*: The uncertainty associated with the outputs returned by a data service should be explicitly modeled, as it is necessary for the interpretation of these outputs by service consumers. The proposed modeling should be compatible with the current Web service standards (e.g., WSDL, SOAP, etc.), as they are widely adopted by Web service development community.

- *Uncertain data service invocation*: We need to define a generic invocation operator which will be able to invoke an uncertain data service and retrieve the confidence degree of its output. An uncertain service may be invoked with both certain and uncertain input data; in the latter case, the invocation

operator should take into account the uncertainty of input to compute the confidence degree of output.

- *Uncertain data services composition*: The conventional service composition model (i.e., the composition algebra and its implementations by different composition execution engines) should be extended to allow for computing the probabilities of the composition's outputs to help users understand and interpret them correctly. An uncertain service should be compose-able with normal and uncertain services alike; i.e., a composition that is unaware of uncertainty should be able to use uncertain services without affecting its normal execution.

4.1.3 Contributions

We summarize below our contributions in this chapter:

- *A probabilistic model for uncertain Data services*: we propose a probabilistic approach to model the uncertainty of outputs returned by an uncertain data service. We extend the service description standard WSDL to accommodate the probabilities of outputs.

- *A probability-aware data service invocation*: We propose an invocation model which allows the invocation of data services with certain and uncertain input. In the first case, the invocation process retrieves the probabilities of the service's outputs. In the second, the invocation process computes the probabilities of returned results based on the probabilities returned by the service and the probability of the input.

- *A composition model for uncertain data services*: We define the semantics of uncertain service composition based on the possible world theory. We propose a probability-aware composition algebra to compute the probabilities of the composition outputs.

The rest of the chapter is organized as follows. In Section 2, we present our probabilistic models for uncertain data services and their invocation. We define

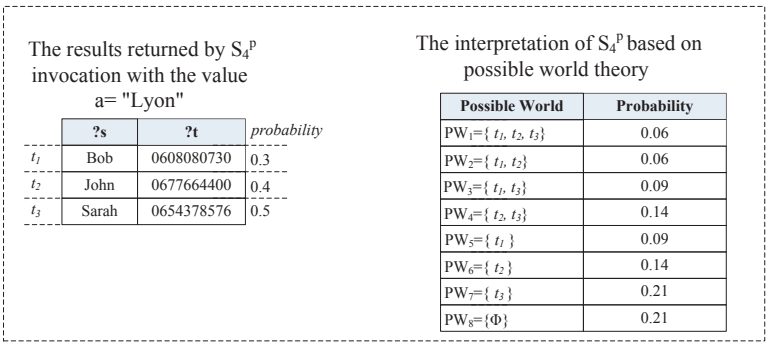
our proposed composition model in Section 3 and we explain our approach for the BID services in Section 4. Finally we conclude the chapter in Section 5.

4.2 Uncertain Data Services: A Probabilistic Model

4.2.1 A description model for uncertain data services

Data uncertainty management has received a considerable attention from the database research community over the last decade. Two main challenges were addressed: uncertainty modeling and query processing over uncertain data. Different approaches were proposed to model data uncertainty [Sadri 1991, Abiteboul 1987, Bosc 2010]. Among these models, the probabilistic and the possibilistic models are the most adopted due to their simplicity. In the probabilistic data model, data uncertainty is modeled as a probability distribution over the possible tuple/attribute values [Marian 2011, Abiteboul 1987]; i.e., each possible tuple/attribute value is assigned a degree of confidence, quantifying its probability. The probabilistic model is a numerical model that relies on an additive assumption and adopts the possible worlds semantics, where an uncertain relation is viewed as a set of possible instances (worlds). Each instance represents the real world with a confidence degree. The structure of these worlds could be governed by underlying generation rules (e.g., mutual exclusion of tuples that represent the same real-world entity). In the possibilistic data model, each possible tuple/attribute value is assigned a (normalized) degree representing how possible?is that value. The possibilistic model is a qualitative, hence a non-additive, uncertainty model.

In this section we give our model for representing uncertain data services. Our model adopts a probabilistic approach to describe the uncertainty associated with data services. In this dissertation we consider that an uncertain service has certain semantic and behavior, only the services can return uncertain results. An



The results returned by S_4^p invocation with the value a= "Lyon"

	?s	?t	probability
t_1	Bob	0608080730	0.3
t_2	John	0677664400	0.4
t_3	Sarah	0654378576	0.5

The interpretation of S_4^p based on possible world theory

Possible World	Probability
$PW_1=\{t_1, t_2, t_3\}$	0.06
$PW_2=\{t_1, t_2\}$	0.06
$PW_3=\{t_1, t_3\}$	0.09
$PW_4=\{t_2, t_3\}$	0.14
$PW_5=\{t_1\}$	0.09
$PW_6=\{t_2\}$	0.14
$PW_7=\{t_3\}$	0.21
$PW_8=\{\Phi\}$	0.21

Figure 4.1: Sample of results returned by S_4^p and the corresponding possible worlds

uncertain service may have one or more operations. Each operation may have one or more output parameters. The uncertainty associated with an operation may have two distinct levels: the individual output parameter and the whole output parameters set levels. These levels correspond to the attribute and the tuple levels in the relational model [Sadri 1991]. At the individual output parameter level, an output parameter may have multiple values from discrete or continuous domains [Marian 2011], and each value has a given probability (that can be estimated by different techniques [Soliman 2010]).

Definition. An uncertain data service is defined as follows:

$$S^p(\overline{I}, \overline{O^p}), \text{ where}$$

- $\overline{I}$ and $\overline{O^p}$ are respectively the input and output parameters vectors of the service S^P.
- p represents the probability associated with output vector $\overline{O}$. p takes a value between 0 and 1 (inclusive).

In this definition the input $\overline{I}$ represents only certain values. A certain service can be viewed as a particular uncertain service with probability $p = 1$.

Example. The service $S_4^p(\$city, ?sales, ?phone)$ in Fig. 4.1 returns the sales representatives along with their phone number in a given city. The service is invoked with city="Lyon". Each of the output tuples t_1, t_2 and t_3 is associated with a

probability p representing the degree of confidence. These probabilities are not part of the output parameters, they are simply metadata provided by the service provider.

The semantics of uncertain data service can be explained based on the possible worlds theory [Sadri 1991]. The probabilistic output tuples returned by the invocation can be interpreted as a set of possible worlds (PW_1,..., PW_n) and each possible world PW_i contains certain tuples and has a probability p_{PW_i} which is dependent on its contained tuples. For example, assuming that the output tuples t_1, t_2 and t_3 returned by S_4^p in Fig. 4.1 are independent probabilistic events, then we obtain eight possible worlds corresponding to the different combinations of tuples. For instance the probability of PW_3 is $0.3 * 0.5 * (1 - (0.4)) = 0.09$, since it contains the tuples t_1 and t_3 and does not contain t_2.

Note that the interpretation of the probabilistic outputs depends on how these outputs are correlated because possible worlds' contents and probabilities depend on that correlation. In this present work we suppose that all returned outputs are independent events.

4.2.2 An Invocation Model For Uncertain Data Services

In this section we analyze the impact of data uncertainty on the service invocation process. Our objective is to define the invocation functionality and give insights on how its semantics should be extended to deal with uncertainty.

Notations. Let S^p be an uncertain data service, I denote certain inputs to the invocation process; I^p denote uncertain inputs: $I^p =< I, P >$, where P denotes the probability of I. Let O denote certain outputs of the invocation process; O^p denote uncertain outputs: $O^p =<O, P>$, where P denotes the probability of O.

Based on the input type (whether it is certain I or uncertain one I^p) we identify the following two invocation classes: conventional invocation and probabilistic invocation. If the input is certain I the invocation is conventional and O^p represents the set of returned outputs $\{O_1^p =< O_1, P_1 >, ..., O_n^p =< O_n, P_n >\}$. The probabilistic invocation refers to the service invocation with uncertain inputs I^p. We use foundations of possible worlds theory to explain the semantics of the probabilistic

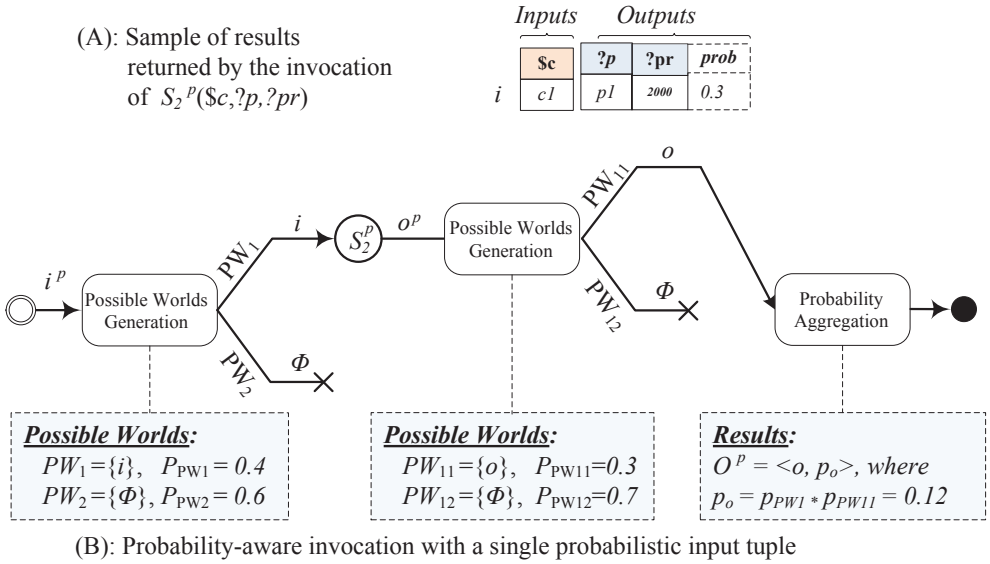


Figure 4.2: Semantics of the probability-aware service invocation

invocation. Fig. 4.2 shows the results of the invocation of the service S_2^p with the input $I^p = i$, where $i =< customer = \text{“}c_1\text{”}, p_i = 0.4 >$
The probabilistic invocation is interpreted as follows: I^p can be represented as a set of possible worlds $\{PW_1, PW_2\}$ as $PW_1 = \{i\}$, and $PW_2 = \{\emptyset\}$. The probability of a possible world is derived from the probabilities of its involved tuples. For example PW_1 contains only one tuple i and thus its probability $p_{PW_1} = Prob(i) = 0.4$; PW_2 does not contain i thus $p_{PW_2} = 1 - Prob(i) = 0.6$. The probability of the output o is calculated as: $p_o = p_{PW_1} * p_{PW_{11}} = 0.4 * 0.3 = 0.12$ such as PW_{11} is a possible world of the interpretation of o and $PW_{11} = \{o\}, PW_{12} = \{\emptyset\}$.

Formally, the uncertain data service invocation can be defined in *extensional* manner (without materializing the possible worlds) as follows:

$$Invoke^p(S^p, I^p) = \{(O_1, P_{O_1} = P_1 * P_i), ..., (O_n, P_{On} = P_n * P_i)\} \qquad (1)$$

4.3 Uncertain Data Service Composition Model

While individual web services can provide interesting information, users queries often require the composition of multiple services. The existing web service com-

position systems [Barhamgi 2013b, Srivastava 2006] don't address the problem of uncertainty. Data uncertainty is an important issue that must be taken into account in composition processes to allow for the right interpretation of returned results. In the following we define the semantics of uncertain services composition.

4.3.1 Composition Semantics

In the case of uncertain data services, the interpretation of a composition is a bit harder that of deterministic services. In this case, we are interested not only in computing the composition's results, but also in their probabilities. For example, assume that the uncertain services S_2^p and S_3^p are involved in a composition to find the products ordered in "Lyon": The table in Fig.4.3(b) shows the results returned by S_3^p (along with their probabilities) when invoked with the value Lyon. The tables (c) and (d) in Fig. 4.3 give the results returned by S_2^p when invoked with the values c1 and c2, respectively. S_3^p returns the tuples t_1, t_2, and t_3 which are independent. These tuples are interpreted into eight possible worlds [Sadri 1991] and the table in Fig. 4.3(c) shows these worlds with their probabilities. Notice that tuples in each world are considered as certain. For example, the world $PW1$ includes the tuples: t_1, t_2, and t_3; and hence the probability of that world is computed as follows: $P_{PW_1} = prob(t1) * prob(t2) * prob(t3) = 0.3 * 0.4 * 0.5 = 0.06$ (we assume that the returned tuples are independent); the probability of $PW2$ is $P_{PW_2} = prob(t1) * prob(t2) * (1 - prob(t3)) = 0.3 * 0.4 * (1 - 0.5) = 0.06$, since that world contains the tuples t_1 and t_2 and does not contain t_3. Fig. 4.3(e) shows the execution plan for the composition in Fig. 4.3(a). For each of the possible worlds corresponding to the results returned by S_3^p (denoted by I^p in the plan in Fig. 4.3(e)), there is an interpretation of the composition, each interpretation has a probability and is represented by a branch in the composition plan. Note that inside each branch we may use the conventional data processing operators (i.e., Projection, Selection, Join, etc.) as exchanged tuples are certain tuples. In each branch, S_2^p is invoked with the tuples of the corresponding world. The invocation operator computes the probability of the outputted tuples by multiplying the probability of the corresponding world with that of the data returned by S_2^p. For instance, the probability of the tuple l_1 outputted (in the first branch) is computed

as follows: $I^p.P_{PW_1} * prob(S_2^p.l1) = 0.06 * 0.3 = 0.018$.

The results returned by the invocation of S_2^p in each branch are probabilistic (and are denoted by ^{p}), and are interpreted as a set of possible worlds. For example, the results $L^p = l_1, l_2, l_3$ returned by S_2^p in the first branch have eight possible worlds. The probabilities of these worlds depend on involved tuples and the considered world of I^p. For example, the probability of the first world in the first branch is computed as follows:
$P = I^p.P_{PW_1} * L^p.P_{PW_1} = 0.06 * [prob(l1) * prob(l2) * prob(l3)] = 0.06 * [0.3 * 0.4 * 0.6] = 0.06 * 0.072$.
That is, a composition corresponds to a set of possible compositions. For instance, in our example we have two services, each has eight possible worlds, hence $n = 8 * 8 = 64$. Each of these compositions may return results different from the other compositions. The same tuple may exist in multiple worlds; for instance the tuple $< p1 >$ exists in the first six worlds of the first branch. The operator Aggregation at the end of each branch computes the probability of each tuple by summing the probabilities of involved worlds. For example, the probability of the tuple $< p1 >$ at the end of first branch is computed as follows:
$p(p1) = \sum P(L^p.PW_i) * P(I^p.PW_1) = (0.072 + 0.048 + 0.108 + 0.072 + 0.168 + 0.112) * 0.06 = 0.0348$
The final aggregation operator computes the probability of tuples across the different worlds corresponding to I^p (i.e., across the different branches). The final probability of $p1$ added all probabilities where $p1$ exists so $p1 = 0.3238 = 0.0348 + 0.027 + 0.027 + 0.0812 + .0.027 + .0.063 + 0.056$.

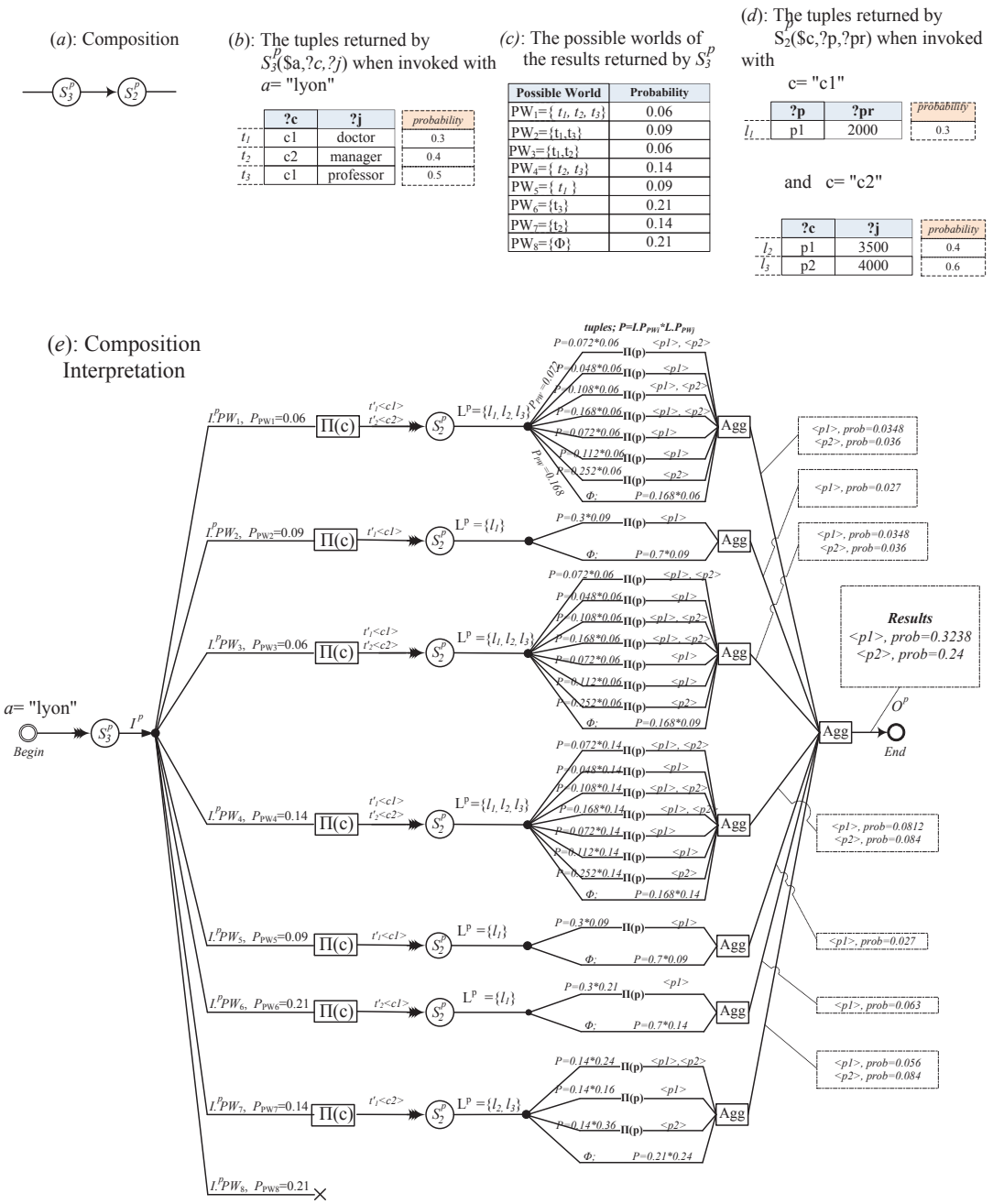


Figure 4.3: Composition and Execution of uncertain Services

4.3.2 An Algebra for Uncertain Data Services Composition

A composition may include multiple probabilistic Web services. When the outputs of these services are aggregated, the probabilities of the obtained results should be computed. These probabilities may be important for many reasons: computing the best results, to assess the quality of results, to take the right decisions, etc. Computing final results' probabilities requires exploring different combinations of possible worlds to assess the composition. Computing all possible worlds after the invocation of each service is ineffective as the number of these worlds is exponential with the number of tuples.
To solve this problem, we opt for an extensional approach (i.e., an approach that does not require the materialization of the possible worlds) and we define a set of composition operators that are needed to formulate the orchestration plans of services compositions [Yu 2008] including probabilistic Web services. These operators assume that the processed tuples are uncorrelated (i.e., the processed tuples are independent from each others).

- $Invoke^p(S^p, I^p)$: The definition of this operator was given in Section 2.

- $Aggregate^p(\overline{I_1^p}, ..., \overline{I_n^p}, \overline{a})$: Let $\overline{I_i^p}$ (where $1 \leq i \leq n$) be a vector of probabilistic tuples outputted by a given service Si, and $\overline{a} a set of attributes; the aggregate operator joins the vectors \overline{I_1^p}, ..., \overline{I_n^p}$ over $\overline{a}$. The probability of an aggregated tuple t is computed as follows: $p(t) = p_{t_I 1} *, ..., p_{t_I i} *, ..., p_{t_I n}$, where $t_{I_i} 1 \leq i \leq n$ are the tuples being aggregated from $Ii (1 \leq i \leq n)$.

- $Project^P(I_i^p, \overline{a})$: Let I_i^p be a vector of probabilistic tuples, and $\overline{a}$ a set of attributes. The project operator projects the vector over $\overline{a}$ and the probability of a tuple t in the outputted set is computed as follows:
$prob(t) = 1 - \prod_{t' : \prod_{\overline{a}}(t') = t} (1 - prob(t'))$

- $Select(\overline{I^p}, \overline{c})$: Let $\overline{c}$ be a set of conditions; The probability of a tuple t in the outputted set is computed as follows: $prob(t) = \begin{cases} prob(t) & \text{if } \overline{c} = true \\ 0 & \text{if } \overline{c} = false \end{cases}$

To answer a given query, uncertain data services must be arranged in an order that depends on their inputs and outputs. However, given a composition of services, different execution plans (a.k.a. orchestrations) may be possible. Not all of these plans are correct; i.e., different plans give different probabilities to the outputted final results. For example in Fig. 4.4 S_3^p and S_2^p are involved in a composition to know the products ordered by the consumers in "Lyon" and the plan is $Project_p^p(Invoke(S_2^p, Invoke(S_3^p, \text{“}Lyon\text{”})))$. We notice that the probability of "p1" is incorrect, it should be 0.3238 as it is calculated using possible worlds' theory in Fig. 4.3. This observation is not surprising as it is already known in the literature that not all queries accept an execution plan that could correctly compute the probabilities. Such queries are called *hard queries* as they have a $\#P-complete$ data complexity under probabilistic semantics [Dalvi 2007].

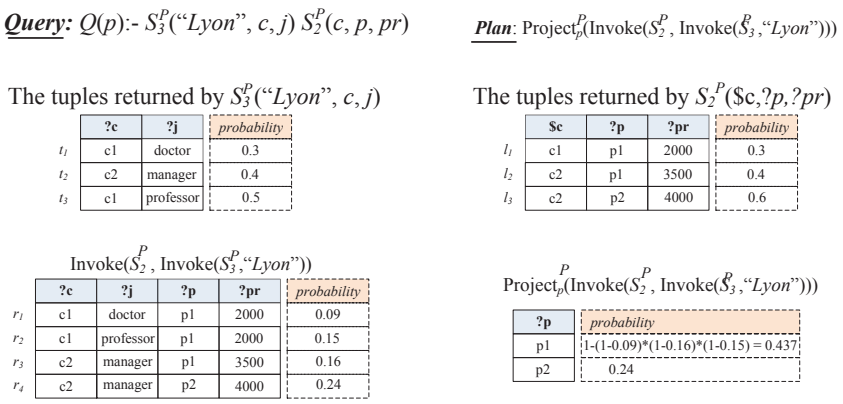


Figure 4.4: Evaluation of $Project_p^p(Invoke(S_2^p, Invoke(S_3^p, \text{“}Lyon\text{”})))$

4.4 Block Independent Disjoint (BID) services

The interpretation of the outputs S^p depends on how these outputs are correlated; i.e., the dependency relationships among output tuples. Even though output tuples may be, in theory, bound by complex correlations, in most application domains these correlations are limited to the mutual exclusion of tuples that map to the same real world entity, e.g., the same school is mapped to multiple addresses, the same individual is mapped to different ages, etc. We therefore limit ourselves in

this work to services whose outputs can be represented by the block independent disjoint model (i.e., BID model) [Suciu 2011] which is suitable for modeling this type of correlations. We call this class of services as BID-Representable P-Services.

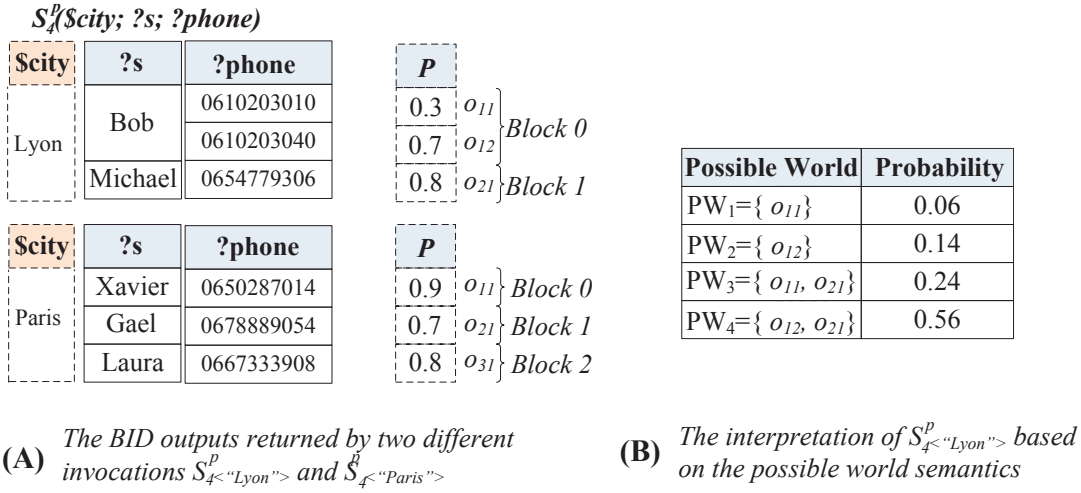


Figure 4.5: (A)Sample of results returned by the BID-P service S_4^p (B)The possible worlds of $S_4^p < "'Lyon"' >$

4.4.1 BID-P-Service Definition

A p-service S is a BID Representable P-Service (BID-P-Service for short) if its output parameters can be partitioned into three classes separated by semi-colons: $(K; A; P)$, where $K = \{K_1, ..., K_n\}$ is called the possible world key, $A = \{A_1, ..., A_n\}$ is the rest of the output parameter set and P is the probability attribute.

Informally, the output results S_i^p returned by the invocation of BID-P-service S^p can be partitioned into blocks, tuples inside each block are disjoint, while tuples from different blocks are independent.

Example. The service $S_4^p(\$city, ?s, ?phone)$ is a BID-P-service since its output parameters can be partitioned into $K = \{s\}$, $A = \{phone\}$, i.e., $O^p = (s, phone)$. Figure.4.5(A) shows $S_4^p <' Lyon' >$ and $S_4^p <' Paris' >$. The result set $S_4^p <' Lyon' >$ involves two blocks, the first one contains two disjoint tuples o_{11} and o_{12}. $S_4^p <' Paris' >$ contains three independent blocks containing one tu-

ple, each. Figure.4.5(B) shows the possible worlds corresponding to $S_4^p <' Lyon' >$.

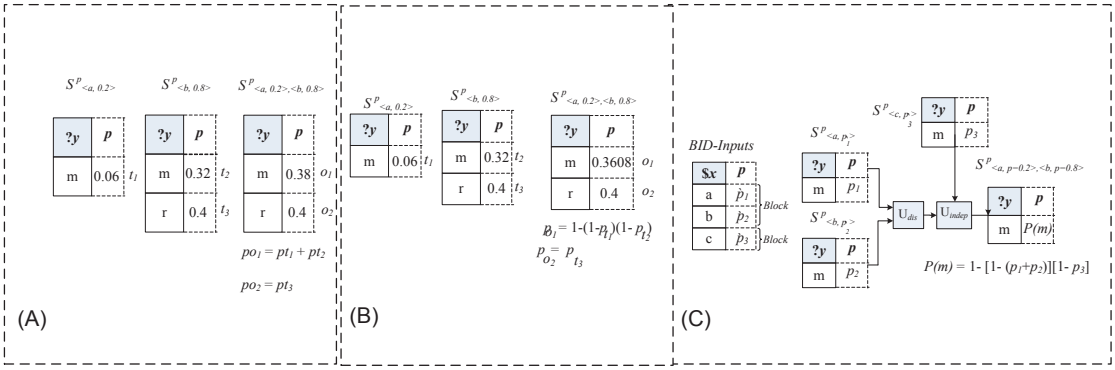


Figure 4.6: Composition and Execution of Uncertain BID Services

4.4.2 BID-P-Service invocation

We define the invocation operators for a set of disjoint input tuples, a set of independent input tuples, and a BID tuples set.

(Case-a) Invocation with disjoint input tuples. We define the semantics of this operator in [Benslimane 2013]. We give below the formula to compute output probabilities without materializing the possible worlds.

$$Invoke^p(S^p_{disjoint}, I^p_j) = \bigsqcup_{disj_{j=1:n}}(S^p, I^p_j) \qquad (2)$$

where $\bigsqcup_{disj}$ computes the probabilities of output tuples (o_i, P_{o_i}) as follows:
$p_{o_i} = \sum P_{l_j} | l_j = o_i \wedge l_j \in \{(S^p, I^p_1), ..., (S^p, I^p_n)\}$
Figure.4.6 gives an example of how *Formula (2)* is applied.

(Case-b) Invocation with independent input tuples. We give below the formula to compute output probabilities.

$$Invoke^p(S^p_{Indep}, I^p_j) = \bigsqcup_{Indep_{j=1:n}}(S^p, I^p_j) \qquad (3)$$

where $\bigsqcup_{Indep}$ computes the probabilities of output tuples as follows:

$$p_{o_i} = 1 - (1 - p_{l_j})(1 - p_{l_k})$$

where $o_i = l_j, l_k...$ and $l_j, l_k... \in \{(S^p, I_1^p), ..., (S^p, I_n^p)\}$
Figure.4.6 gives an example of how *Formula (3)* is applied.

(Case-c) Invocation BID input. based on Formulas (2) and (3), the Formula in the case of BID input is:

$$Invoke^p(S^p_{BID}, I_1^p, ..., I_n^p) = \bigsqcup_{BID_{j=1:n}}(S^p, I_j^p) \qquad (4)$$

where $\bigsqcup_{BID}$ is equivalent to $\bigsqcup_{disj}$ when the aggregated tuples come from invocations that belong to the same input block, and is equivalent to $\bigsqcup_{Indep}$ when the aggregated tuples come from invocations that belong to different input blocks.
Figure.4.6 gives an example of how *Formula (4)* is applied.

4.4.3 BID-P-Service composition

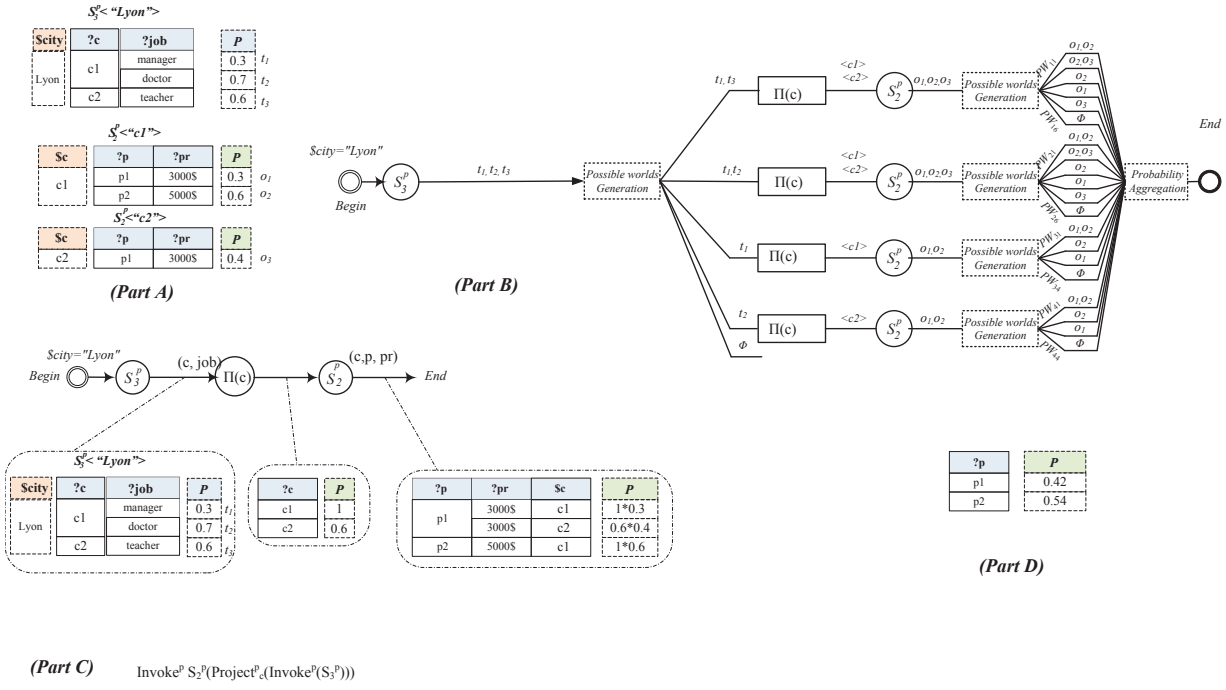


Figure 4.7: (A)Invocation of $S_3^p(\$a, ?c, ?j)$ and $S_2^p(\$c, ?p, ?pr)$(B)The interpretation of a composition (C)The composition Plan of Q_1 (D)The results

In this section we intuitively define the semantics of uncertain service composition through an example.
Consider the services $S_3^p(\$a, ?c, ?j)$ and $S_2^p(\$c, ?p, ?pr)$ whose invocation results are shown in figure.4.7. The composition of S_3^p and S_2^p to answer a query Q_1:
$Q_1(p) : -S_3^p(\$a, ?c, ?j)S_2^p(\$c, ?p, ?pr)$
S_3^p is invoked with the value "$Lyon$" and returned the tuples $\{t1, t2, t3\}$. Then S_2^p is invoked obtained tuples in each branch are, in turn, interpreted into possible worlds represented as sub branches. For example the branch involving the possible worlds PW_1 and PW_{11} constitutes a possible composition PC_1 whose probability is simply the product of the probabilities of involved possible worlds: $P_{PC_1} = P_{PW_1} * P_{PW_{11}} = 0.18 * 0.108 = 0.01944$. In our example we have 20 (6+6+4+4) different possible compositions. The probability of a tuple o in the composition result is the sum of the probabilities of all possible compositions that return o. For example the tuple o_1 is returned by the possible compositions: PC_1, PC_2, PC_7, PC_{10}, PC_{13}, PC_{15}, PC_{17} and PC_{19} so $probability(o_1) = 0.01944+0.03024+0.04536+0.03024+0.0216+0.0144+0.0504+0.0336 = 0.24528$.
Another approach to compute the composition results is to express the composition plan using the operators presented in section 4.3.2: $Invoke^p(S_2^2)(Project_c^p(Invoke^p(S_3^p)))$. The figure.4.7 shows how we compute the probabilities of both intermediate and the output tuples. The probabilities of output results are correct as they are equal to those obtained using the possible worlds semantics. However, not all composition plans expressed in that algebra give the correct probabilities.

4.5 Conclusion

In this chapter, we proposed a probabilistic approach for modeling uncertain data services for two cases: independent data and Block Independent Disjoint data. Specifically, we showed how the uncertainty associated with a data service can be modeled, and proposed a composition algebra (i.e., a set of operators) that can compute the probabilities of the outputs of a composition. Last but not least, we integrated our model within an existing composition system.

Chapter 5

Safe Plan

Contents

5.1 Introduction 70

5.1.1 Motivating scenario 70

5.1.2 Challenges 71

5.1.3 Contribution 72

5.2 Background 72

5.3 Safe composition for data services with independent tuples 74

5.3.1 Example 74

5.3.2 Criteria for safe composition plans 76

5.4 Safe composition for data services with BID tuples 78

5.5 Conclusion 79

5.1 Introduction

A composition may accept different execution composition plans expressed all with the probabilistic algebra. Not all of these plans compute correctly the probabilities of outputs. The objective of this chapter is to define the conditions under which a plan returns the correct probabilities, and in which case we call it a *safe composition plan.*

5.1.1 Motivating scenario

Assume we have two data services: $S_1(\$city; ?school; ?zip; ?reputation)$ returns the schools (along with their zip codes and reputations) in a given city - input parameters are proceeded by \$ and the output ones by ?. $S_2(\$school; ?course; ?teacher)$ returns courses (and their teachers) that are taught at a given school. These services are uncertain services as they integrate open Web databases (e.g., S_2 integrates open databases from *nces:ed:gov*[1] and *psk12:com*)[2]. Assume a student, Alice, is looking for the best math courses taught in her city, Washington. Alice expects that the best math courses are those taught in highly rated schools. Therefore, she invokes S_1 with the value $city =$ "*DC*", then she selects highly rated schools, and invokes S_2 with their names to get their proposed math courses. The mashup shown in figure 5.1 implements the following query:

$$Q_1(course, school, teacher) :$$
$$-S_1(\text{“}DC\text{”}, ?school, ?zip, \text{“}high\text{”}), S_2(\$school, ?course, ?teacher)$$

We assume in this example that S_1 and S_2 could return, in addition to their outputs, metadata information representing the probabilities of their returned uncertain output data. For example, S_1 returns two schools “Lincoln" and “Heritage" with different combinations of their reputation and zip code, each combination is associated with a probability. Now, if the mashup plan has computed its output with ignoring the probability metadata, then the order of outputted math courses

[1] The National Center for Education Statistics

[2] The premier source for school performance information about public elementary, middle and high schools

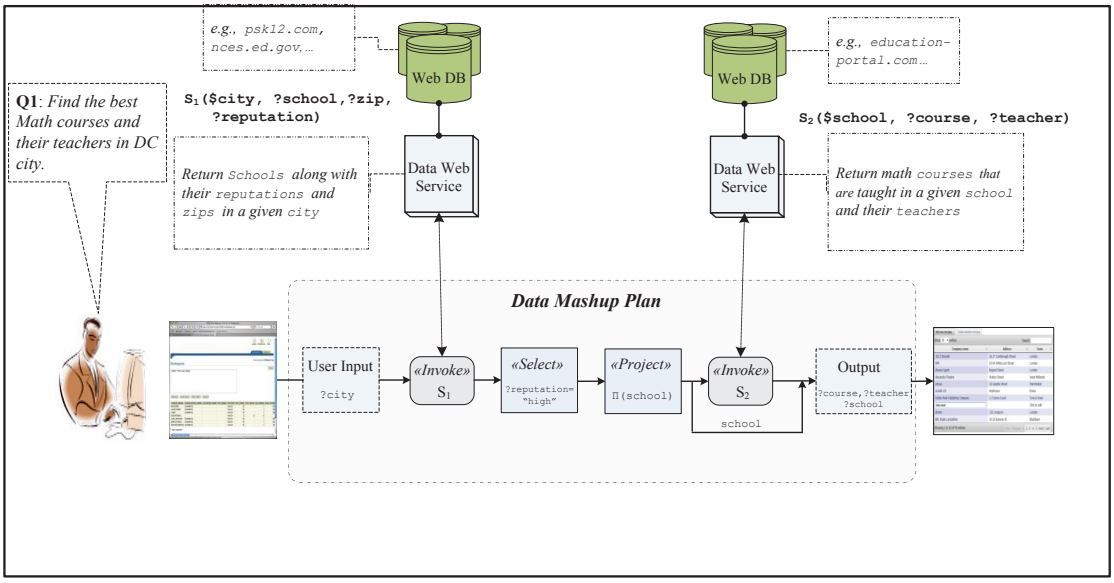


Figure 5.1: An Example of a composition

would be "Co10"' then "Co12", the same order of S_2's output. In contrast, if the probability metadata is considered in ordering the mashup's output, then "Co12" would appear before "Co10", as its probability is higher than that of "Co10". The importance of considering the uncertainty (i.e., probability) of output data becomes clearer when the mashup's output includes a sheer number of math courses, where the most probable ones may not appear first, leading users to miss the most interesting results among the complete results list.

5.1.2 Challenges

As explained in the running example, it is important to compute the probabilities of the query results to understand and use them correctly, e.g., the probabilities of output tuples in the example were used to rank them and retain the most probable results. The problem of query evaluation over probabilistic databases has been receiving an increasing attention from the database research community. A fundamental result in the field is the classification of queries in two types: *easy* and *hard* queries. Easy queries can be evaluated efficiently using the probabilistic algebra, i.e., they accept a plan (called a *safe* plan) that computes the probabilities

correctly. On the other hand, the plans of hard queries are all unsafe. Hard queries are often evaluated using some intensional probabilistic inference techniques which are known to be hard and quite inefficient.

5.1.3 Contribution

In this chapter, we propose some conditions to check if the composition plan is safe. The main contributions of this chapter are as follows:

- We define the notion of a safe orchestration plan which is a query plan P that can be evaluated using extensional semantics on a one instance; in contrast, the standard definition of a safe plan is one where the extensional semantics is correct on any instance.
- We define a safe orchestration query plan in the case of independent tuples. We propose a set of conditions to satisfy the safety of the plan.
- We define a safe orchestration query plan in the case of BID tuples.

The remainder of this chapter is organized as follows. In Section 2, we present a background of safe plan. In section 3 we introduce our safe composition for data services with independent tuples and in the section 4 we show the safe composition for BID services. Finally, in section 4, we summarize our contributions and conclude.

5.2 Background

A probabilistic service $S^p = (S, p)$ represents a probability distribution over outputs set of S. The outputs of S are modeled as possible worlds also called instances [Bosc 2010]. The evaluation of a Boolean query q on a probabilistic set D of probabilistic services is defined by Pr(q), which is the sum of probabilities of those instances of D that satisfy q. In this thesis we study efficient techniques for evaluating q.
Suppose the input relations to an operator are independent. Then, we define the *extensional semantics* of the relational operators as follows:

- $Invoke^p(S^p, I^p)$: it represents the invocation operator. It invokes S^p with the input I^p.
- $Aggregate^p(\overline{I_1^p}, ..., \overline{I_n^p}, \overline{a})$: it represents the join operator. It aggregates I_1^p,..,I_n^p according to the attribute a.
- $Project^P(I_i^p, \overline{a})$: it represents the projection operator. It projects I_i^p according to the attribute a.
- $Select(\overline{I^p}, \overline{c})$ it represents the selection operator. It selects I^p which check the condition c.

Extensional operators can be computed efficiently, and they return what looks like a representation of an independent relation. Suppose we take the output of an extensional plan, and interpret it as an independent probabilistic relation. If this probabilistic relation is the same as the possible worlds semantics then we say that the plan is *safe*:

Definition 1. Let P be a query plan and D a probabilistic database instance. A plan P is called *safe* if its extensional semantics is equal to the possible worlds semantics.

Definition 2. [Jha 2010] define safe plan as follows:
Consider a probabilistic database instance D and a query plan P. Let o be an operator in P. A set T of input tuples to o is called set of offending tuples if o becomes safe after removing the tuples in T.

The set of offending tuples do not necessarily come from the database instance; they could also be intermediate tuples generated during the query plan. So many tuples in the database instance, that make the query unsafe, may actually correspond to just one offending tuple for the query plan. Note that a safe plan has no offending tuples and the output of any plan is an expression with symbols from only the offending tuples. Hence the number of offending tuples is a measure of how safe/unsafe a plan really is for a given database instance.

5.3 Safe composition for data services with independent tuples

In this section we focus on the issue of the orchestration issue of p-services participating in a composition. We show, through examples, that not all composition plans give correct probabilities. We define then a set of criteria under which a composition plan computes the correct probabilities. We assume that the services to be composed are already identified (either automatically by one of the systems in [Benaouret 2011][Srivastava 2006][Barhamgi 2013a][Sabesan 2009], or manually by users). All compositions considered in our discussion answer Select-Project-Join (SPJ) queries.

5.3.1 Example

Consider the services $S_1^p(\$city; ?school; ?zip; ?reputation)$ and $S_2^p(\$school; ?course; ?teacher)$whose invocation results are shown in figure.5.2. Assume a query Q_1 for the best math courses in Washington:

$$Q_1(x; y; z) : -S_1^p(\text{“}DC\text{”}; ?y; ?l; \text{“}high\text{”})S_2^p(\$y; ?x; ?z)$$

Composition interpretation based on the possible semantics: The composition of S_1^p and S_2^p to answer Q_1 can be interpreted as follows (figure.5.2(B)): S_1^p is invoked with the value “DC" and returned the tuples set $\{t_1, t_2, t_3\}$. Only the tuples with a high value for the reputation attribute are retained. This set is interpreted into 8 possible worlds, each world corresponds to a branch in figure.5.2(B). The probability of each world is calculated based on the probabilities of the tuples belonging to this world. For example $P_{PW1} = t_3 = (1 - P_{t_1}) * (1 - P_{t_2}) * P_{t_3}$= (1-0.2)*(1-0.7)*0.1=0.8*0.3*0.1=0.024.
Then, we project the results on “school" attribute and we invoke the service S_2. For each world we generate possible worlds. For example for the possible world PW_1 we generate 2 possible worlds: $\{PW_{11} = (o_3), PW_{12} = (\emptyset)\}$.
Finally, we calculate the math teachers' probabilities by the sum of the probabil-

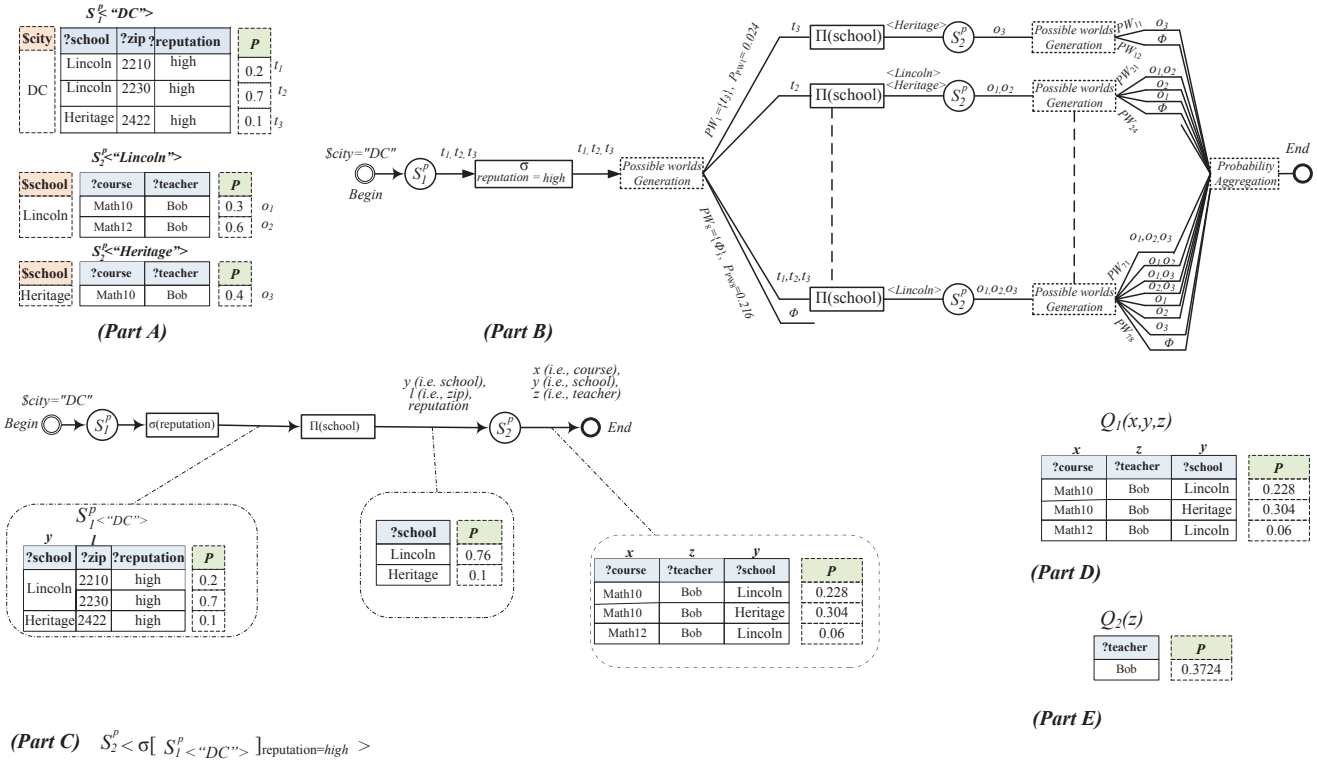


Figure 5.2: Composition Plan

ities of all possible worlds where the teacher belongs and the result is in 5.2 (E): $P(Bob) = 0.3724$.

Evaluation with an extensional execution plan: Figure.5.2 (C) shows a plan for Q_1, S_1^p is invoked with the deterministic value (city = "DC"). Only the tuples with high reputation are retained. These tuples are then projected on the school attribute, and the results are used to invoke S_2^p. The figure shows also how we compute the probabilities of both intermediate and the output tuples. The probabilities of output results are correct as they are equal to those obtained using the possible worlds semantics (figure.5.2 (D)).

Now consider a second query Q_2 in which the mashup user is interested in knowing the best math teachers in a given city (e.g., the mashup user may be looking for good private math teachers). S_1^p and S_2^p can be composed to answer Q_2, as the best math teachers are those who are teaching in schools with high reputations.

Q_2 can be expressed as that of Q_1 except we project the final result on z which returns only one tuple $o =< Bob, 0.495 >$ where the probability of o is computed as follows:$prob(o) = 1_(1 - 0.228) * (1 - 0.304) * (1 - 0.06) = 0.495$. This probability is not equal to the probability obtained using the possible world semantics (0.3724). This observation is not surprising as it is already known in the literature that not all queries accept an execution plan (called safe plan) that could correctly compute the probabilities [Dalvi 2007]. Such queries are called hard queries as they have a #P-complete data complexity under probabilistic semantics [Dalvi 2007]. However, the hard queries given in the literature do not commonly arise in practice. For example, only 20% of the TPC/H benchmark queries (www.tpc.org) fall in this category.

5.3.2 Criteria for safe composition plans

A safe composition plan is guaranteed to compute all output probabilities correctly. We define bellow a set of conditions under which a composition plan is safe. We call such compositions as safe compositions. We start by defining the dependency graph of a composition.

Definition. (Dependency Graph G): The dependency graph G of a composition is a directed acyclic graph in which nodes correspond to services and edges correspond to dependency constraints between component services. We say that there is a dependency constraint between two services S_i and S_j (S_j depends on S_i) if one of S_i's output parameters is an input parameter of S_j.

Safe composition plan p**.** We say that p is safe if:

1. p respects G,
2. all edges in p are joins that involve the primary key of at least one probabilistic service,
3. p is tree,

4. a probabilistic service appears in p at most once,

5. the primary keys of services that are leaves in p appear at the p's output.

Examples.

- The plan of Q_1 that is shown in (E) satisfies our conditions thus it is safe. The one of Q_2 violates the condition 5, thus is unsafe.

- We suppose that we have another probabilistic service S_3^p which returns students in a given city along their level. Assume a query Q_3 to know the best math teachers of the level 7 in "Lincoln" school. To answer Q_3, S_2^p and S_3^p can be composed and the plan is as follows:
  P: $Project^p_{teacher}(Invoke^p(S_2^p(\text{“}DC\text{”})), Invoke^p(S_3^p(\text{“}NY\text{”})))$.

  However, the two services don't have any dependency so moreover the first condition is violated that's why the plan P is unsafe.

- Assume a query Q_3 to know the best schools in Washington and New York and the plan is as follows:

$$P :$$
$$Project^p_{school}(select^p[Invoke^p(S_1^p(\text{“}DC\text{”}))]_{reputation=high}[Invoke^p(S_1^p(\text{“}NY\text{”}))]_{reputation=high}).$$

  This plan P is unsafe because the service S_1^p appears twice so it violates the fourth condition.

- Assume a query Q_4 to know the students taking courses with the best math teachers. Q_4 can be expressed as that of Q_1 except we invoke in the last the service S_3^p but in this case the plan will not be safe because it violates the first and the last condition.

- We suppose that the service S_1^p returns another attribute which is "address" we have another probabilistic service S_4^p which returns apartments for rent

along their price in a given address. Assume a query Q_5 to know the apartments which their price lower than 40000$ and near to the best school in “NY". To answer Q_5 we opt for the following plan:

$$P : Project^p_{Apart}[Aggregate^p[Select^p(Invoke^p(S_1^p(\text{“}NY\text{”}))_{reputation=high},\\ Select^p(Invoke^p(S_4^p(\text{“}Broklyn\text{”})))_{price<40000}]]$$

We notice that P violates the condition 5 thus it is unsafe.

5.4 Safe composition for data services with BID tuples

In this section we focus on the orchestration issue of p-services with BID tuples. We show, through the same example of the previous section, that even in the case of BID tuples not all composition plans give correct probabilities. We define then a set of criteria under which a composition plan computes the correct probabilities in this case.

The composition of S_1^p and S_2^p to answer Q_1 can be interpreted as follows (figure.5.3(B)): S_1^p is invoked with the value “DC" and returned the tuples set $\{t_1, t_2, t_3, t_4\}$. This set is interpreted into 6 possible worlds.

Figure.5.3 (C) shows a plan for Q_1, S_1^p is invoked with the deterministic value (city = “DC") and returns a BID table. Only the tuples with high reputation are retained. These tuples are then projected on the school attribute, and the results are used to invoke S_2^p. The figure shows also how we compute the probabilities of both intermediate and the output tuples. The probabilities of output results are correct as they are equal to those obtained using the possible worlds semantics (figure.5.3 (D)).

For the query Q_2, the same as that of independent tuples, the probability according to the plan $\prod_z^p Q_1$ is equal to 0.7746 which is different from the probability obtained using the possible world semantics in 5.3 (E). We can check the same set of conditions defined for the independent tuples to check if an orchestration of

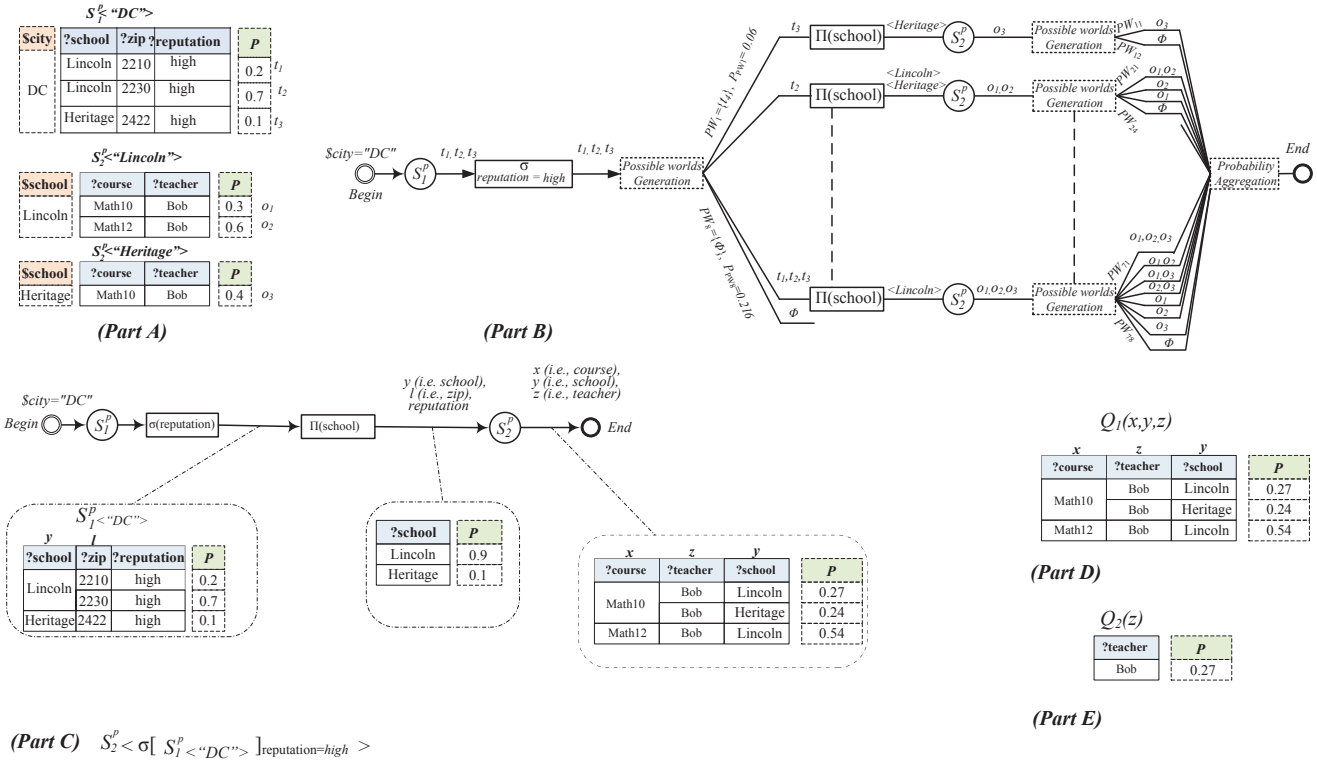


Figure 5.3: BID Composition Plan

BID-P services is safe. As the case of independent tuples, the plan of Q_1 satisfies our conditions thus it is safe. The one of Q_2 violates the condition 5.

As the case of independent tuples, the plan of Q_1 satisfies our conditions thus it is safe. The one of Q_2 violates the condition 5.

5.5 Conclusion

Finding efficient safe orchestration plans is vital for query evaluation over probabilistic services. In this chapter, we studied through examples the safety of orchestration plans in two cases: services with independent tuples and with BID tuples. Moreover, we proposed a set of conditions that should be checked to verify the safety of plans on both of these two cases.

Chapter 6

Implementation and evaluation

Contents

6.1 Introduction 81

6.2 Prototype 81

6.2.1 Architecture 81

6.2.2 Service Annotation 83

6.2.3 Service Invocation 85

6.2.4 Query formulation 85

6.2.5 Query rewriting 86

6.2.6 Composition Execution 86

6.3 Implementation and experimental results 88

6.3.1 Technical environment 88

6.3.2 Preference-Aware Query Model 89

6.3.3 Uncertainty in Web Services Composition 93

6.3.4 Experimental results 93

6.4 Conclusion 96

6.1 Introduction

This chapter is devoted to the implementation and performance study of our proposed approach for uncertain data web services composition. We implemented our different techniques and applied them to the real-estate and e-commerce domains. We provide in this chapter a performance study of our composition framework.

The remainder of this chapter is organized as follows. In Section 2, we present the architecture of our implemented system. In section 3 we provide the technical environment and the experimental results. Finally, in section 4, we summarize our contributions and conclude the chapter.

6.2 Prototype

6.2.1 Architecture

The architecture of our implemented system for querying and composing uncertain data services is shown in Figure 6.1. The architecture is organized into four layers. The first layer contains a set of Oracle/MySQL databases that store the data. The second layer includes a set of proprietary applications developed in Java; each application accesses databases from the first layer (i.e. it executes parameterized queries over the databases). These proprietary applications are exported as uncertain data web services to the system. These services constitute the third layer. We used the deployment kit bundled with the GlassFish Web server to build and deploy our data Web services over a set of GlassFish Web servers running on top of set of PC machines (running Windows XP).

The description files (i.e. WSDLs) of data Web services in the third layer are annotated with RDF views that describe their semantics from the perspective of RDFS domain ontologies. Annotated description files are published to Web service registries. The upper layer includes a Graphical User Interface (GUI) and our composition system. Users access the system via a GUI implemented using Swing, the widget toolkit for Java. They can submit specific or parameterized queries to the composition system.

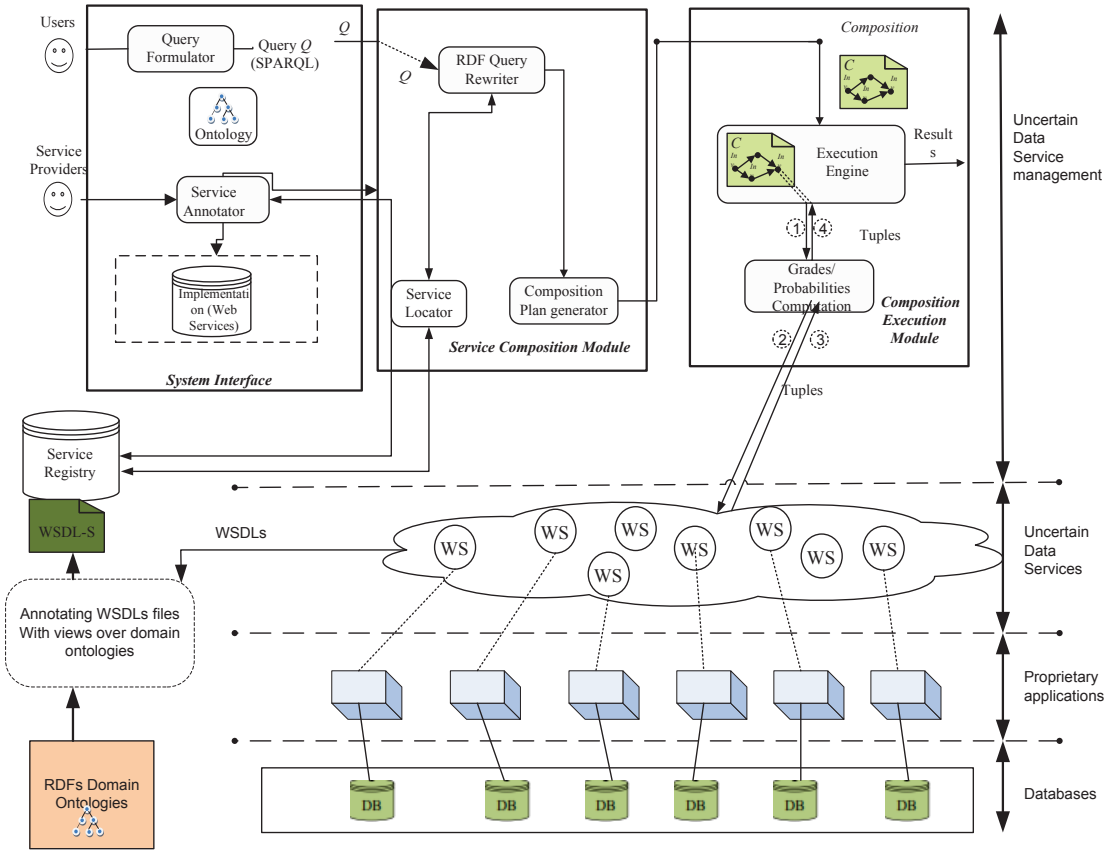


Figure 6.1: System architecture

The Interactive Query Formulator helps users specify their RDF queries (SPARQL queries) over the mediated ontology in an interactive manner. The Service Locator retrieves WSDL-S service descriptions of relevant services by accessing the service registries (UDDI registries). The RDF Query Rewriter implements our RDF query rewriting algorithms; it determines whether the services returned from the Service Locator can be used for answering the posed query; if services can be composed to answer the query it outputs possible compositions. The Composition Plan Generator generates execution plans for the obtained compositions. The generated plans are either sent for immediate execution or are deployed as new Web services; the choice depends on whether the posed query is specific or parameterized. In case of a parameterized query, the composition plan generator will also

generate a WSDL description file for the generated plan. The execution engine implements the different operators used in the generated plans. The exchanged messaged will be transformed with the invoked services if needed. Since services can have schemas for their input and output different from schemas corresponding to views, service providers need to specify the mapping between the input message and the xml schema obtained from the serialization of input parameters of the associated RDF view.

6.2.2 Service Annotation

```
<interface name="Schools">
    <operation name="schoolByCountry" pattern=wsdl:in-out
       wssem:modelReference="RDFSOntology:School"
       wssem:modelReference="RDFSOntology:Country">
       <!-RDF View is added as extensible element on an operation -->
       <rdfannot:rdfquery name="query" value="Select ?S
       ?P.rdf:type.O:School
       ?S.O:hasprice.$p
       ?S.O:hasrep.$r
       ?S.O:hastuition.$t
       ...      />

....
    </operation>
</interface>
```

Figure 6.2: A Portion of a WSDL File Annotated with RDF Views

In the WSDL-S [1], inputs, outputs and operations can be annotated with concepts from domain ontologies to capture their semantics using the extensibility feature of WSDL. WSDL-S proposal defines a new attribute called *modelReference* to associate input and output messages and the operations with the corresponding ontological concepts. In our work, we follow the same approach to associate the services' operations with their corresponding *RDF views*. To do so, for each operation element we define a new element ***rdfquery*** to link each operation with

[1] http://www.w3.org/Submission/WSDL-S/

its RDV view. Figure 6.2 shows a part of a WSDL file annotated with RDF views.

```
<?xml version="1.0" encoding="UTF-8"?>
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
<S:Body>
<ns2:outputMessage xmlns:ns2="http://org.me/">
<tuple probability ="0.8" grade="0.76" key="S">
<Product>p1</Product>
<S>s1</S>
<r>good</r>
<t>45000$</t>
<a>Lyon</a>
</tuple>
</ns2:outputMessage>
</S:Body>
</S:Envelope>
```

Figure 6.3: An example of a SOAP message

To correctly use an uncertain service, the probabilities, the grades and the correlations of its outputs should be modeled and integrated into service description standards.

Probability and grade inclusion in Web service standards. We extended the Web service standards (WSDL, SOAP) to take into account our model of uncertain services. WSDL 2.0 is the last official W3C recommendation for Web services description. WSDL 2.0 defines several extensibility elements that can be used to annotate the service descriptions files with metadata and semantic information. These elements can either be added to attributes or to XML elements of the service description, the main requirement being that the extensibility elements are defined in their own namespace. We exploited these elements of WSDL2.0 and defined the following three attributes on the `output message` elements: "`probability`" to specify the probability degree associated with each output element (i.e., tuple), "`grade`" to define the matching degree relative to users' preferences and "`Key`" to specify that an output parameter plays the role of an identifier (i.e. a primary key) - recall that identifier attributes are needed for computing the correct plan of a composition. Figure 6.3 shows an example of a SOAP message annotated with

the different attributes (“grade", “probability", “key").

6.2.3 Service Invocation

The composition system [Barhamgi 2013b, Barhamghi 2010] relies on a standard Java API for Web services invocation JAX-WS (jax-ws.java.net). This Java API allows SOA application developers to call and consume Web services in their applications. Specifically, the *Dispatch* interface (*javax.xml.ws.Dispatch*) allows invoking a service by constructing/reading the service's input/output (XML) messages. It enables the developers to work on the XML message level by either constructing the invocation messages manually using the desired XML API (e.g. JDOM, etc), or by using the Java Architecture for XML Binding (JAXP jaxb.java.net/) to translate between XML messages and internal Java objects that constitute the SOA application. Fig. 6.4 shows how we extended this API to implement our invocation model. The input I^p has the form of a Java object (the probability is simply a filed in the corresponding Java class) and is the argument of the whole invocation process. Then, an input XML invocation message will be constructed; this process can be done manually using an XML API or automatically based on JAXB Java/XML mappings. The obtained message is then encapsulated by an SOAP envelope and sent to the Web service. Then, the SOAP message returned by the service is de-encapsulated to extract the output XML message. The latter is then read; if the output XML message is read manually by an XML API, the code should then read the value of the *Probability* and *grade* attributes in the WSDL service description, otherwise the *Probability* (*grade*) attribute should be mapped to the probability (grade) field of the Java object by the JAXB Java/XML mappings. Finally, the Probability (grade) will be updated by taking into account the probability (grade) of the input.

6.2.4 Query formulation

The Interactive Query Formulator helps users specify their RDF queries (SPARQL queries) over the mediated ontology in an interactive manner. Users formulate their queries over domain ontologies in SPARQL query language.

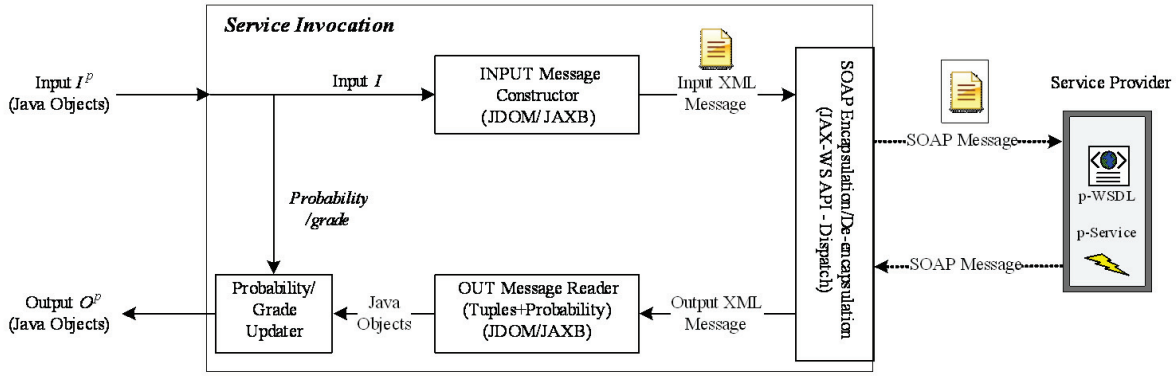


Figure 6.4: Implementation of uncertain Invocation based on the JAX-WS API

6.2.5 Query rewriting

The component RDF Query Rewriter is the heart of our architecture in Figure 6.1; it allows rewriting the received SPARQL query in terms of available data Web services. This component was implemented in Java. The implementation contains three distinct Java packages:

- The package parsing: contains a set of classes for reading SPARQL queries.
- The package sparqlquery: contains a set of classes for representing SPARQL quires.
- The package soqr (Service Oriented Query Rewriting): ontains a set of classes for rewriting a SPARQL query in terms of Web services and for generating the execution plans of the corresponding compositions.

6.2.6 Composition Execution

We have elected to devise our own composition language and implement our own execution engine. Our composition language features the data processing operations in Table 6.1. Our implemented execution engine can execute each of the operations in Table 6.1 and allows for data streaming between the different operations used in a uncertain data service composition. The execution engine is

Table 6.1: Data processing operations

Operation	Descriptions
InvokeThread	A thread-based process invoking the service with each data tuples in the input relation and returns the output along the grade of the probability.
JoinThread	A thread-based process joining the tuples in a set of relations and calculates the new grades/probabilities.
SelectThread	A process selecting the tuples that satisfy a given condition in a given relation.
UnionThread	A thread-based process unifying the tuples in a set of relations.

implemented by a Java package called execution that contains the following main classes:

- Composition: This class represents the composition plan.
- InvocationThread: This class allows invoking a service within an independent thread.
- JoinThread : This class allows to join the outputs of two or more data services. The join is done within an Independent thread.
- UnionThread : This class allows to combine the outputs of multiple similar web services and eliminates data redundancy.
- ProjectionThread : This class allows to project the desired attributes from a data tuple.
- SelectionThread : This class allows filtering tuples based on data values.

6.3 Implementation and experimental results

6.3.1 Technical environment

The development phase is divided into two parts. In the first, we used two types of web services: those physically deployed on an server application and those created locally. To develop an approach based on Web services, different Java middleware exist such as Aparche Axis, JBoss and Glassfish with the features and benefits of their own. We chose Glassfish for the following reasons:

- Development environment and tools are fully integrated in the NetBeans IDE.
- Compliance with specifications Web services and interoperability standards.
- Open-source project with a strong industrial support from both Sun Microsystems and Microsoft.

Thereafter, we define the development environment and libraries needed to implement our framework:

- IDE editor: Netbeans 6.9.
- Processes Intel (R) Core (TM) i5 - 4GB RAM.
- Web services platform: Glassfish 3.
- ava Web services API: JAX-WS.

6.3.2 Preference-Aware Query Model

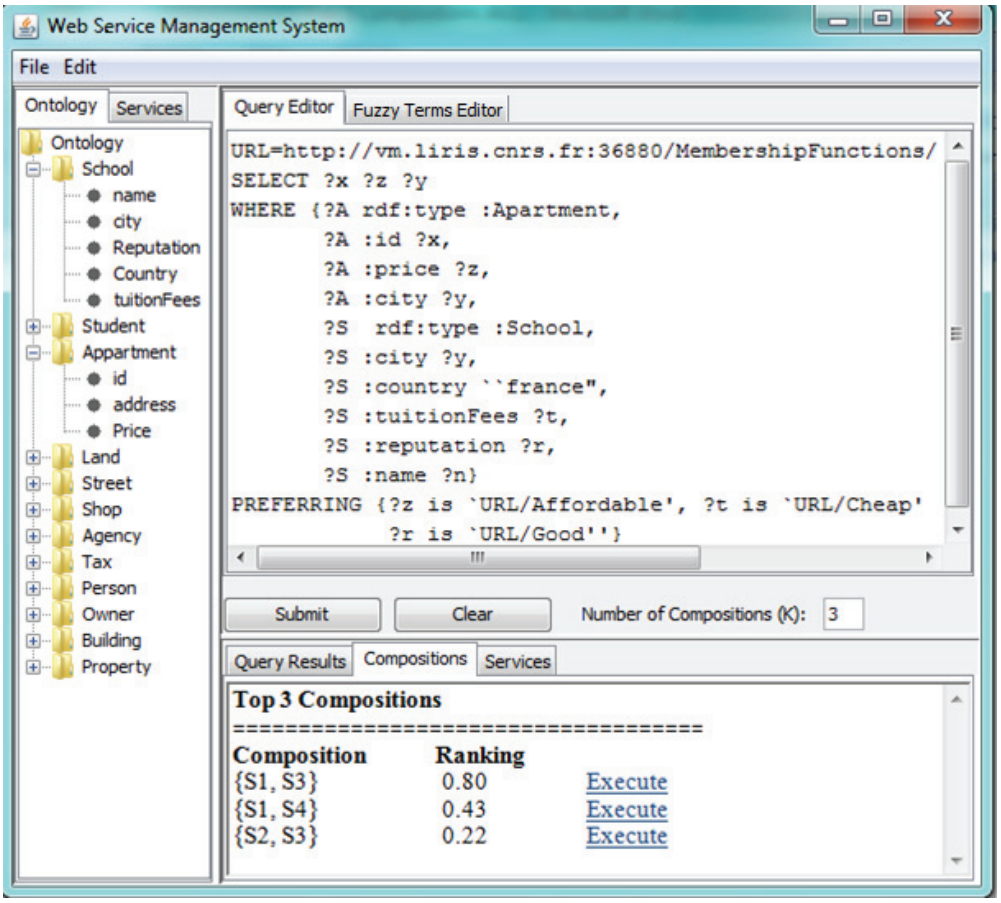


Figure 6.5: The preference query formulator interface

To evaluate and validate our approach, we implemented all of the components shown in figure6.1 in Java. The ranking-aware composition execution engine was implemented to allow for both scalar and vector grades computations and with any of the three T_Z, T_P, and T_L norms.

We conducted a series of experiments to evaluate the efficacy of our approach. The experiments covered many queries from the real estate domain with a rich set of fuzzy preferences over a set of services returning synthetic data about Apartments, Lands, Restaurants, etc. Our experiments shown that the overhead incurred by computing the rankings is negligible compared to the time necessary to execute the same generated compositions without any ranking at all. In

addition, the returned top-k tuples were always correct, proving the soundness of our proposed operators.

Figure 6.5 shows the preference query formulator interface. The user uses this interface to enter his/her sparql query with fuzzy preferences. This query is formulated over an existing ontology. The user can execute the query and chose any of the displayed compositions. On the left-hand side, the panel Ontology presents a tree-like view of domain ontology, the panel Services presents the services stored inside service registries. The Query Editor on the right-left side is space where users edit their queries. SPARQL savvy users can express their queries directly in the Query Editor of our interface.

Fuzzy terms are those stored in the fuzzy terms knowledge base of our system. Users can edit and test them via the interface in figure 6.6 to identify the relevant fuzzy terms. Users can also define their own fuzzy terms.

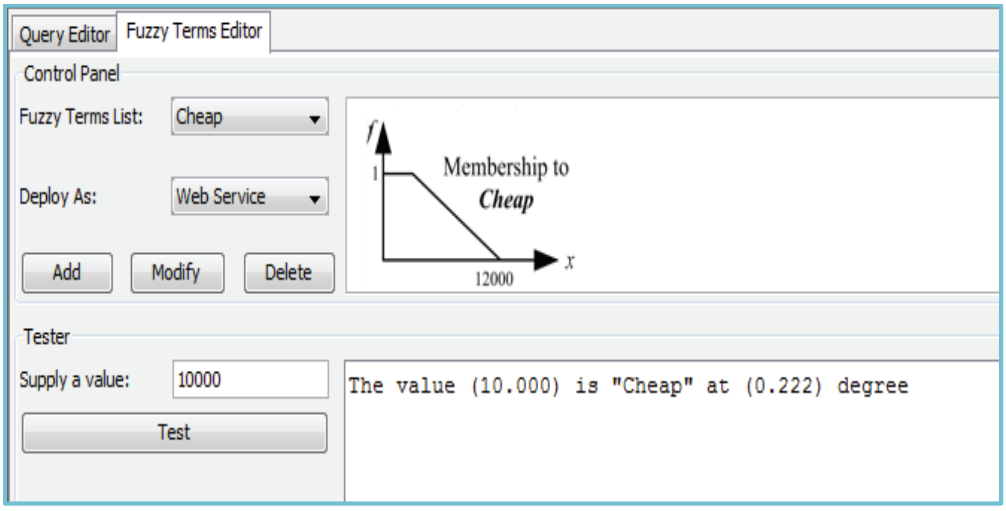


Figure 6.6: The fuzzy term editing

The composition execution plan is then displayed on figure 6.7 and the user is allowed to choose an execution strategy. If the user wants to aggregate the grades of its fuzzy preferences, he/she chooses the scalar grades computing; otherwise if he/she wants to keep an eye on the grades of all of its fuzzy preferences, he/she chooses the vector grades computing. The user has also to set his/her strategy: optimistic (T_Z norm), reinforcement (T_P norm), and pessimistic (T_L norm).

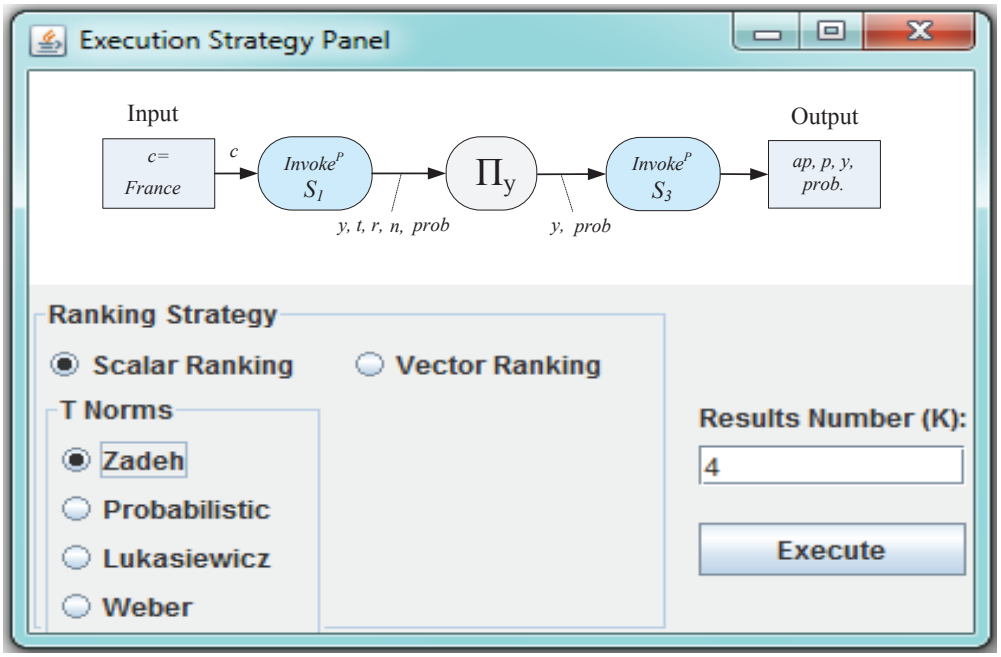


Figure 6.7: The execution strategy panel

Scalar Ranked results are displayed in figure 6.8. The results are ordered by their aggregated grades.

In figure 6.9, results are ranked according to their vectors of grades (each grade corresponds to a degree of satisfaction of a fuzzy user preference). To do so, we make use of the leximin ordering which leads to a total order. This ordering is borrowed from the multicriteria decision field [Dubois 1990].

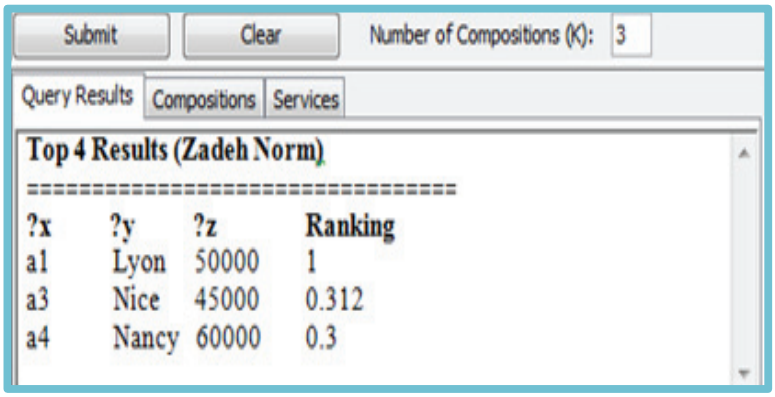


Figure 6.8: The scalar ranked results panel

In case of the user is not satisfied by the obtained query results, he/she can choose another service composition and execute it. In case of empty (resp. too few) results, users can relax (by introducing some tolerance) their fuzzy constraints present in the initial query. The relaxation operation allows for enlarging the support of the membership functions associated with each constraint, thus making the query less selective. It is worthy to note that this operation requires to re-execute the grades computation step. For example, relaxing the affordable constraint of Q will return more results: some of the tuples previously ranked to 0 in scalar grades, or ranked to 0 in all dimensions in vector grades.

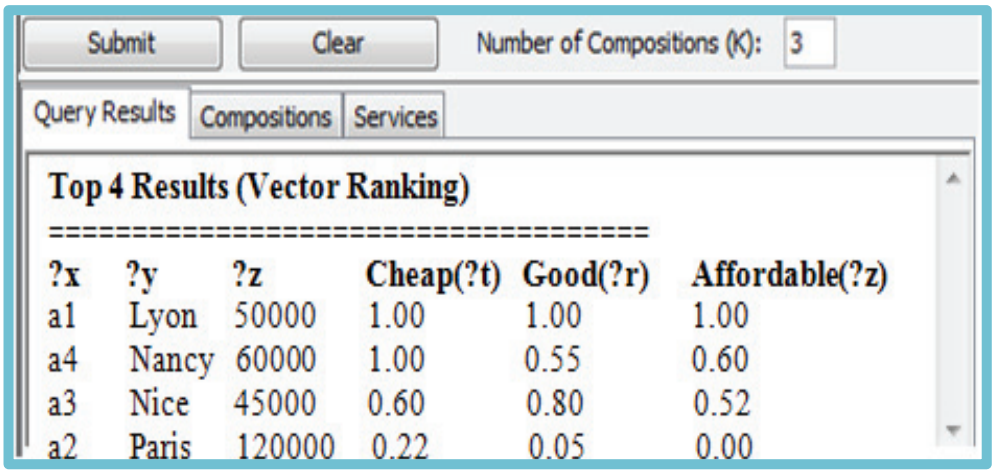


Figure 6.9: The vector ranked results panel

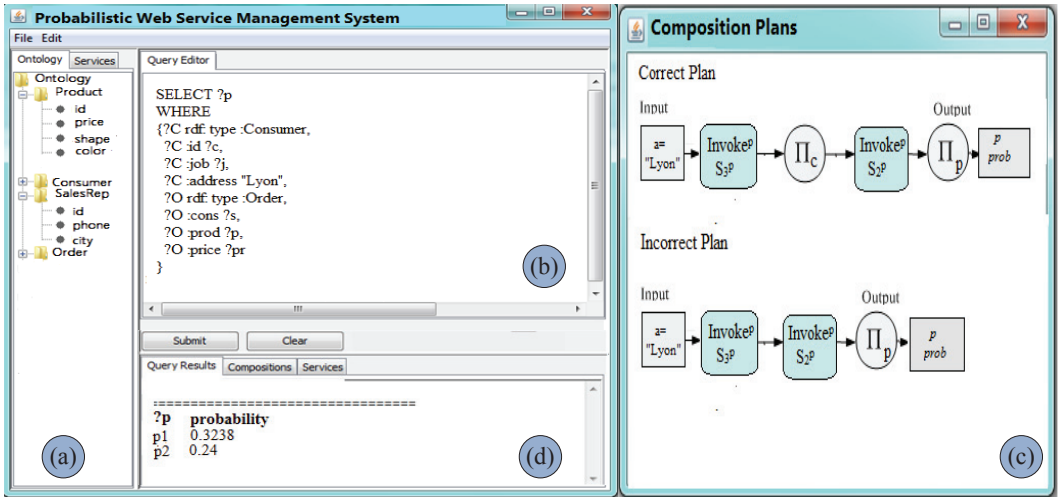


Figure 6.10: Uncertain composition system interface

6.3.3 Uncertainty in Web Services Composition

We implemented the operators of our probability-aware composition algebra (defined in Section 3.2) within the composition engine of [Barhamgi 2013b, Barhamghi 2010] and extended that engine with our algorithm for computing the correct composition execution plan.

Figure 6.10 shows the user interface to our extended service composition system. The panel (*a*) shows domain ontology. The user edits his SPARQL query in the panel *b*. Panels (d) the obtained results along with their probabilities.
Figure 6.10 (c) shows the composition plans and indicates whether the plan is safe or not.

6.3.4 Experimental results

Composition system with fuzzy preferences. Due to the limited availability of real data services, we implemented a Web service generator. The generator takes as input a set of (real-life) model data services (each representing a class of services) and their associated fuzzy constraints and produces for each model service a set of synthetic data services and their associated synthetic fuzzy constraints. The generated data services satisfy some fuzzy constraints on the attributes of the implemented model service.

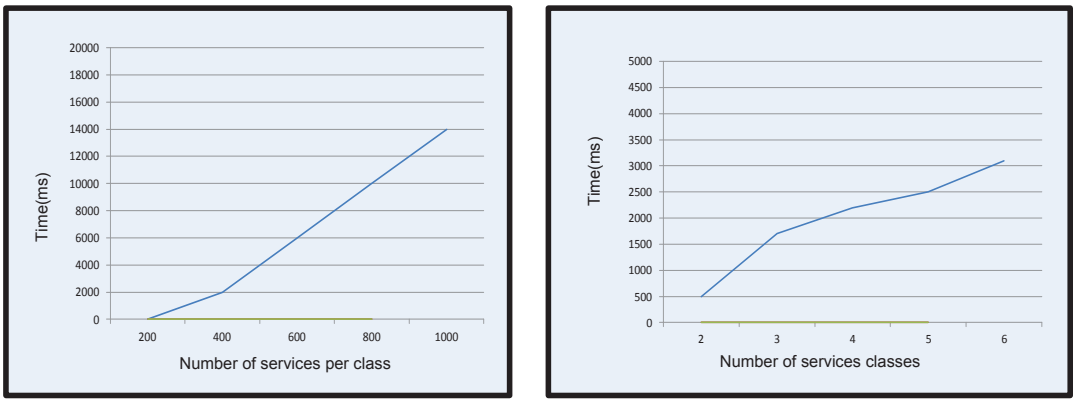


Figure 6.11: Performance results

We measured the average execution time required to solve the composition problem as the number of data services per class increases. We varied the number of data services per class from 200 to 1000. The results of this experiment are presented in Figure. The results show that our framework can handle hundreds of services in a reasonable time. The results of this experiment are presented in figure 6.11 (a).

We measured the average execution time required to solve the composition problem as the number of service classes increases. We varied the classes number from 1 to 6. The results of this experiment in figure 6.11 (b).

We also made a comparison at runtime level between the normal composition and composition with grades calculation and we noticed that the difference is minimal. It becomes apparent that the execution time dedicated to the calculation of grade is negligible. Figure 6.12 shows the results.

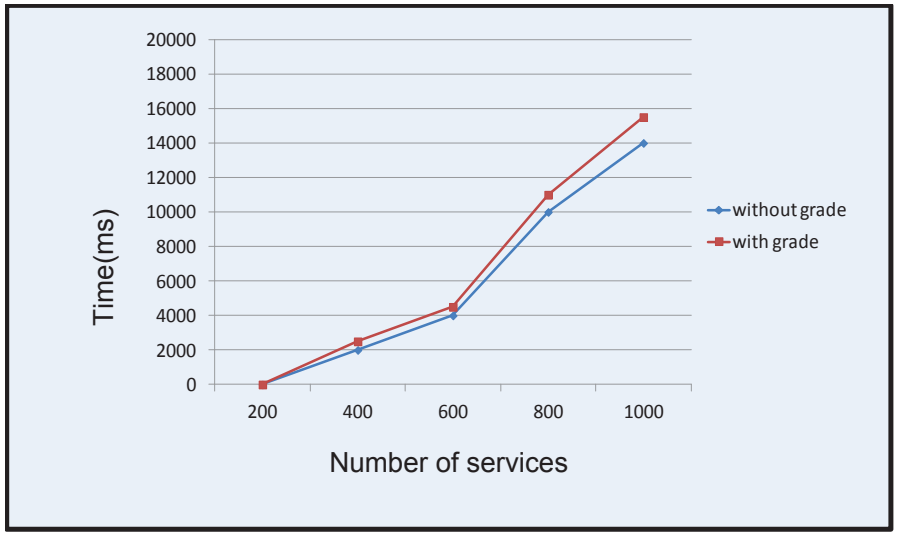


Figure 6.12: Performance results: measuring execution time with and without grades calculation

Composition system with uncertainty. We conducted a series of experiments for two main objectives. First, we wanted to verify how much of real-life service compositions accept correct composition execution plans. Second, we wanted to evaluate the cost incurred by the calculation of probabilities in our composition algebra (relative to the initial composition algebra of the system in [Barhamghi 2010]). For this purpose, we have implemented web services on top of an uncertain database storing synthetic data about products, consumers, sales representatives, etc. This database has a size of 1000MB and simulates the data of the TPC-H benchmark (*www.tpc.org*). The obtained initial results gave the following facts. First, 8 out of 10 real life compositions (i.e., queries) accepted correct execution plans, thus computing the correct probabilities for results. The considered compositions answer queries that are considered as common by the TPC-H benchmark. Second, for our second objective, we measured the execution times for a composition with and without the calculation of probabilities. The results in figure 6.13 show that the time incurred by the calculation of probabilities is negligible.

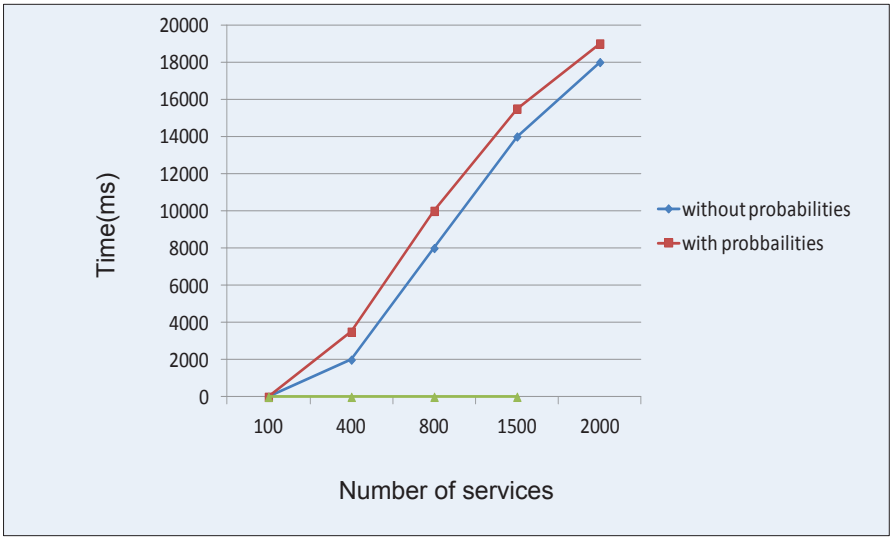


Figure 6.13: Performance results: measuring execution time with and without probabilities calculation

6.4 Conclusion

In this chapter, we have presented briefly the system we have implemented to evaluate our approach for data service composition with uncertainty. We also conducted a performance analysis on a wide data set to assess the efficiency of our proposal. The results showed that our system can handle hundreds of data Web services in a reasonable time even with grades and probabilities calculation.

Chapter 7

Conclusion

Contents

7.1 Conclusions 98

7.2 Future works 99

In this chapter, we summarize the results of our dissertation and discuss future research directions for uncertain data Web service composition.

7.1 Conclusions

In this dissertation, we addressed the uncertainty issues of data Web services and their composition. First, we proposed an approach to answer preferences queries over data Web services. Our approach allows us to improve the descriptions of data services by associating them with fuzzy constraints that better characterize their accessed data. Second, we addressed the uncertainty that may be associated with the services' accessed data by proposing a probabilistic modelling for services. This dissertation covers the different aspects of the above problem, starting from modelling uncertain data services, services selection and composition, to ranking the output results of compositions. We summarize below our major contributions:

- ***Data services composition with user fuzzy preferences.*** We presented an approach for composing Web services while taking into account the user's fuzzy preferences. We proposed a model for data services based on RDF views over domain ontologies. Our model characterizes also the service's accessed data with fuzzy constraints. In our approach, services that match the best with users' preferences (which are also modelled as fuzzy constraints), are selected, then orchestrated within a composition plan that better answers the fuzzy query. We proposed an algebra to orchestrate the selected data services. The proposed algebra ranks the returned results based on their relevance to user's fuzzy preferences.

- ***A probabilistic model for uncertain data services*** We proposed a probabilistic approach to model the uncertainty of the outputs returned by an uncertain data service. The model assumes that an uncertain data service has certain semantics and behaviour. Only its returned results are uncertain. We proposed an invocation model which allows the invocation of data services with certain and uncertain input. In the first case, the invocation process retrieves the probabilities of the service's outputs. In the second, the

invocation process computes the probabilities of returned results based on the probabilities returned by the service and the probability of the input.

- ***A composition model for uncertain data services***. We defined the semantics of uncertain service composition based on the possible world theory [Bosc 2010]. Computing the probabilities of a composition's output based on the possible world theory is inefficient as the number of the possible worlds is exponential with the number of tuples. Thus, we opted for an extensional approach and proposed a probability-aware composition algebra to compute the probabilities of the composition outputs. These probabilities are important for computing the best results, assessing the quality of results, taking the right decisions, etc.

- ***Safe orchestration plan.*** We showed that not all composition plans compute correctly the output probabilities. We studied through examples the safety of orchestration plans in two cases: independent tuples and BID tuples. Moreover, we proposed a set of conditions that should be met to verify the plan's safety in these two cases.

- ***Implementation and performance study.*** We presented the system we have implemented to evaluate our approach for data service composition under uncertainty. We also conducted a performance analysis on a wide data set to assess the efficiency of our proposal. The results showed that our system can handle hundreds of data Web services in a reasonable time even with grades and probabilities calculation.

7.2 Future works

This dissertation leads to various fertile grounds for future researches. We identify the following main directions for future works:

- ***Probability-aware optimization of services composition*** A composition of data services may accept different plans that all respect its dependency graph. Some of these plans compute the correct probabilities while

others do not. These plans have different evaluation costs that could depend on the order of their different operators (e.g., invocations, selections, joins, etc.) as well as on services (e.g., the service selectivity, i.e., the average number of output tuples per one input tuple, its ability to be invoked with blocks of tuples, etc.). More research efforts are needed to study the problem of inferring the best composition plan that still correctly computes the outputs' probabilities. In some applications like Web objects ranking, the most important is to efficiently rank objects (based on their probabilities) rather than to know their exact probabilities. Therefore, an unsafe, but efficient, composition plan that would compute approximate probabilities (but precise enough for the ranking purpose), would be sometimes preferred over a safe, but inefficient, composition plan. Therefore more research efforts are needed to quantify the probability error bounds that could be produced by an unsafe composition plan. The same research goal is beneficial to hard compositions (i.e., compositions that do not accept a safe plan).

- ***Ranking uncertain output data*** data services (or their composition) often returns an overwhelming number of results (e.g., data tuples), thus leading data consumers to miss the ones that are most relevant to their needs. Top-k queries are a common approach to report the best k answers (of a query) based on matching the processed tuples to users' preferences. In the context of uncertain data services, the outputted tuples should be ranked based not only on their matching degrees with users' preferences, but also on their probabilities, and the probabilities of correlated intermediate tuples. Tuple scores and uncertainty interplay to decide the top-k outputted data. The interaction between data uncertainty and the "top-k" gives rise to different possible interpretations of uncertain top-k queries: (i) the "top-k" tuples in the "most probable" world; (ii) the "most probable top-k" tuples that belong to valid possible world(s); (iii) the set of "most probable top ith" tuples across all possible worlds, where i = 1...k, etc. More research efforts are needed to devise new efficient ranking methods and techniques that implement these interpretations and view the data's probabilities as an important ranking dimension. Efforts are also needed to optimize the execution of compositions

answering top-k queries in such a way to stop the composition execution as soon as top-k answers are produced.

- ***The consideration of other modelling approaches of uncertain data.*** In this dissertation we assumed that data providers adopt a probabilistic approach for modelling data uncertainty; i.e., services provide data items and their probabilities. However, this assumption may not always hold true. Some data providers may adopt other approaches to quantify the uncertainty. For example, in the sensors application domain the possibilistic approach may be more convenient; i.e., (uncertain) services provide in this case data items and their possibilities. While in such cases, our service description model remain always reusable (e.g., with replacing the probability information by the possibility information), the invocation and the composition uncertainty calculus become invalid. More research efforts are needed to redefine the invocation models and to devise new uncertainty calculus when several uncertain services are aggregated to answer queries.

Appendix A

Appendix: Academic Achievements

---

1. **Soumaya Amdouni**, Mahmoud Barhamgi, Djamal Benslimane, Rim Faiz: Handling Uncertainty in Data Services Composition. IEEE SCC 2014: 653-660.

2. **Soumaya Amdouni**, Mahmoud Barhamgi, Djamal Benslimane, Rim Faiz, Kokou Yetongnon: Web Services Composition in the Presence of Uncertainty. ER 2014: 136-143.

3. **Soumaya Amdouni**, Djamal Benslimane, Mahmoud Barhamgi, Allel HadjAli, Rim Faiz, Parisa Ghodous: A Preference-Aware Query Model for Data Web Services. ER 2012: 409-422.

4. **Soumaya Amdouni**, Mahmoud Barhamgi, Djamal Benslimane, Allel HadjAli, Karim Benouaret, Rim Faiz: Answering Fuzzy Preference Queries over Data Web Services. ICWE 2012: 456-460.

Bibliography

[Abbaci 2011] Katia Abbaci, Fernando Lemos, Allel Hadjali, Daniela Grigori, Ludovic Liétard, Daniel Rocacher et Mokrane Bouzeghoub. *A Cooperative Answering Approach to Fuzzy Preferences Queries in Service Discovery*. In 9th International Conference, FQAS 2011, Ghent, Belgium, October 26-28, 2011 Proceedings, pages 318–329, 2011. (Cited on page 14.)

[Abiteboul 1987] Serge Abiteboul, Paris Kanellakis et Gosta Grahne. *On the Representation and Querying of Sets of Possible Worlds*. In Proc. ACM SIGMOD, 1987. (Cited on pages 18, 21 et 56.)

[Agrawal 2010] Parag Agrawal, Anish Das Sarma, Jeffrey D. Ullman et Jennifer Widom. *Foundations of Uncertain-Data Integration*. In 36th international conference on VLDB, pages 1080–1090, 2010. (Cited on pages 17, 26, 27 et 52.)

[Alonso 2004] Gustavo Alonso, Fabio Casati, Harumi A. Kuno et Vijay Machiraju. Web services - concepts, architectures and applications. Data-Centric Systems and Applications. Springer, 2004. (Cited on page 2.)

[Andritsos 2006] Periklis Andritsos, Ariel Fuxman et Renee J. Miller. *lean answers over dirty databases: A probabilistic approach*. In 22nd Int. Conf. on Data Eng., 2006. (Cited on page 28.)

[Balbiani 2009] Philippe Balbiani, Fahima Cheikh Alili, Pierre-Cyrille Héam et Olga Kouchnarenko. *Composition of Services with Constraints*. In Formal Aspects of Component Software (FACS 2009), Eindhoven, Pays-Bas, pages 31–46, 2009. (Cited on page 27.)

[Barhamghi 2010] Mahmoud Barhamghi, Djamal Benslimane et Zakaria Maamar. *A Query Rewriting Approach for Web Service Composition.* IEEE Transactions on Services Computing, vol. 3, pages 206–222, 2010. (Cited on pages 5, 27, 36, 38, 49, 85, 93 et 95.)

[Barhamgi 2013a] Mahmoud Barhamgi et Djamal Benslimane. *A privacy aware data service composition system.* In 16th International Conference on Extending Database Technology, pages 57–60, 2013. (Cited on page 74.)

[Barhamgi 2013b] Mahmoud Barhamgi, Djamal Benslimane, Youssef Amghar Nora Cuppens-Boulahia et Frederic Cuppens. *PrivComp: a privacy-aware data service composition system.* In 16th International Conference on Extending Database Technology, pages 757–760, 2013. (Cited on pages 60, 85 et 93.)

[Benaouret 2011] Karim Benaouret, Djamal Benslimane et Mahmoud Barhamgi and. *A web service composition system based on fuzzy dominance for preference query answering.* PVLDB, vol. 12, pages 1430–1433, 2011. (Cited on page 74.)

[Benslimane 2013] Djamal Benslimane et Mahmoud Barhamgi. Uncertain web services. Available upon an email request, 2013. (Cited on page 66.)

[Bosc 2010] Patrick Bosc et Olivier Pivert. *Modeling and querying uncertain relational databases: A survey of approaches based on the possible worlds semantics.* International Journal of Uncertainty Fuzziness and Knowledge Based Systems, vol. 18, pages 565–603, 2010. (Cited on pages 5, 18, 56, 72 et 99.)

[Buneman 2006] Peter Buneman, Adriane Chapman et James Cheney. *Provenance management in curated databases.* In SIGMOD '06 Proceedings of the 2006 ACM SIGMOD international conference on Management of data, pages 539–550, 2006. (Cited on pages 17, 27 et 53.)

[Carey 2007] Michael J. Carey. *Declarative Data Services: This Is Your Data on SOA.* In IEEE International Conference on Service-Oriented Computing and Applications Newport Beach California USA, pages 695–705, 2007. (Cited on pages 2, 12, 14 et 27.)

[Carey 2008] Michael J. Carey. *SOA What?* IEEE Computer, vol. 41, pages 92–94, 2008. (Cited on page 31.)

[Cheng 2004] Reynold Cheng, Dmitri V. Kalashnikov et Sunil Prabhakar. *Querying imprecise data in moving object environments.* IEEE Trans. Knowl. Data Eng., vol. 16, 2004. (Cited on pages 17 et 53.)

[Cheng 2007] Reynold Cheng, Dmitri V. Kalashnikov et Sunil Prabhakar. *Evaluation of Probabilistic Queries over Imprecise Data in Constantly-Evolving Environments.* Inf. Syst., vol. 31, pages 104–130, 2007. (Cited on pages 18 et 21.)

[Chomicki 2003] Jan Chomicki. *Preference formulas in relational queries.* ACM Transaction Database System, vol. 28, no. 4, pages 427–466, 2003. (Cited on page 14.)

[Cormode 2009] Graham Cormode, Feifei Li et Ke Yi. *Semantics of ranking queries for probabilistic data and expected ranks.* In International Conference on Data Engineering ICDE, 2009. (Cited on page 22.)

[Curbera 2002] Francisco Curbera, Matthew Duftler, Rania Khalaf, William Nagy, Nirmal Mukhi et Sanjiva Weerawarana. *Unraveling the Web Services Web: An Introduction to SOAP, WSDL, and UDDI.* IEEE Internet Computing, vol. 6, no. 2, pages 86–93, 2002. (Cited on pages 2 et 11.)

[Dalvi 2004] Nilesh Dalvi et Dan Suciu. *Efficient Query Evaluation on Probabilistic Databases.* In 30th VLDB Conference,Toronto, Canada, pages 864–875, 2004. (Cited on pages 21, 22 et 28.)

[Dalvi 2007] Nilesh Dalvi et Dan Suciu. *Efficient Query Evaluation on Probabilistic Databases.* VLDB Journal., vol. 16, pages 523–544, 2007. (Cited on pages 18, 64 et 76.)

[Dalvi 2011] Nilesh Dalvi, Christopher Re et Dan Suciu. *Queries and Materialized Views on Probabilistic Databases.* Journal of Computer and System Sciences, vol. 77, pages 473–490, 2011. (Cited on pages 19 et 26.)

[Dan 2007] Asit Dan, Robert Johnson et Ali Arsanjani. *Information as a Service: Modeling and Realization.* In International Conference on Software Engi-

neering (Workshop on Systems Development in SOA Environments), 2007. (Cited on page 12.)

[de Calmes 2003] Martine de Calmes, Didier Dubois, Eyke Hullermeier, Henri Prade et Florence SÃ¨des. *Flexibility and fuzzy case-based evaluation in querying.* An illustration in an experimental setting. Int. J. of Uncertainty, Fuzziness and Knowledge-Based Systems, vol. 11, pages 43–66, 2003. (Cited on page 15.)

[de Calmes 2007] Martine de Calmes, Henri Prade et Florence Sedes. *Flexible Querying of Semi-Structured Data: a Fuzzy Set-Based Approach.* International Journal of Intelligent Systems, vol. 7, pages 723–737, 2007. (Cited on page 15.)

[Dong 2009] Xin Luna Dong et Alon Y. Halevy. *Data integration with uncertainty.* VLDB Journal, vol. 18, pages 469–500, 2009. (Cited on page 26.)

[Dubois 1990] Didier Dubois et Henri Prade. *Beyond min agregation in multicriteria decision : (ordered) weighted mean, discri-min, leximin.* Kluwer Publishers., pages 181–192, 1990. (Cited on pages 5 et 91.)

[Dubois 1996] Didier Dubois et Henri Prade. *Using fuzzy sets in flexible querying: Why and how?* In Proc. of the 1996 Workshop on Flexible Query-Answering Systems, pages 89–103, 1996. (Cited on page 15.)

[Dubois 1999] Didier Dubois, Henri M. Prade et James C. Bezdek. *Fuzzy Sets in Approximate Reasoning and Information Systems.* Kluwer Publishers, 1999. (Cited on page 42.)

[Dubois 2000] Didier Dubois et Henri Prade. *Fundamentals of Fuzzy Sets.* The Handbooks of Fuzzy Sets Series Kluwer Boston Mass., vol. 7, 2000. (Cited on page 31.)

[Dustdar 2012] Schahram Dustdar, Reinhard Pichler, Vadim Savenkov et Hong-Linh Truong. *Quality-aware service-oriented data integration: requirements, state of the art and open challenges.* ACM SIGMOD Record, vol. 41, pages 11–19, 2012. (Cited on page 52.)

[Eid 2008] Mohamad A. Eid, Atif Alamri et Abdulmotaleb El-Saddik. *A reference model for dynamic web service composition systems*. IJWGS, vol. 4, no. 2, pages 149–168, 2008. (Cited on page 13.)

[Fagin 2003] Ronald Fagin, Amnon Lotem et Moni Naor. *Optimal Aggregation Algorithms for Middleware.* J. Comput. Syst. Sci., vol. 66, pages 614–656, 2003. (Cited on pages 23 et 25.)

[Fuhr 1997] Norbert Fuhr et Thomas Rolleke. *A probabilistic relational algebra for the integration of information retrieval and database systems.* ACM Transaction., 1997. (Cited on pages 19, 21 et 22.)

[Fung 2012] Benjamin C.M. Fung, Thomas Trojer, Innsbruck Patrick C.K. Hungand Oshawa Li Xiong, Atlanta Khalil Al-Hussaeni et Rachida Dssouli. *Service-Oriented Architecture for High-Dimensional Private Data Mashup.* IEEE T. Services Computing, vol. 5, pages 373–386, 2012. (Cited on page 53.)

[Gatterbauer 2010] Wolfgang Gatterbauer, Abhay K. Jha et Dan Suciu. *Dissociation and propagation for efficient query evaluation over probabilistic databases*. In Workshop on Management of Uncertain Data, 2010. (Cited on page 28.)

[Gilpin 2007] Mike Gilpin, Noel Yuhanna, Katie Smillie, Gene Leganza, Randy Heffner et Jost Hoppermann. *Information-As-A-Service: Waht's Behind This Hot New Trend?* Forrester Research, 2007. (Cited on page 12.)

[Green 2006] Todd J. Green et Val Tannen. *Models for Incomplete and Probabilistic Information*. In Current Trends in Database Technology EDBT 2006, pages 278–296, 2006. (Cited on page 18.)

[Guinard 2010] Dominique Guinard, Vlad Trifa, Stamatis Karnouskos, Patrik Spiess et Domnic Savio. *Interacting with the SOA-Based Internet of Things: Discovery, Query, Selection, and On-Demand Provisioning of Web Services.* IEEE T. Services Computing., vol. 3, pages 223–235, 2010. (Cited on page 31.)

[Gursky 2008] Peter Gursky, Veronika VanekovÃ¡ et Jana Pribolová. *Fuzzy user preference model for top-k search.* In FUZZ-IEEE 2008. (IEEE World Congress on Computational Intelligence), pages 1606–1612, 2008. (Cited on page 15.)

[Hadjali 2008] Allel Hadjali, Souhila Kaci et Henri Prade. *Database Preferences Queries A Possibilistic Logic Approach with symbolic Priorities.* In FoIKS, pages 291–310, 2008. (Cited on pages 26 et 31.)

[Hadjali 2011] Allel Hadjali, Souhila Kaci et Henri Prade. *Database preference queries - a possibilistic logic approach with symbolic priorities*. Ann. Math. Artif. Intell, vol. 63, no. 3-4, pages 357–383, 2011. (Cited on page 14.)

[Halpern 1990] Joseph Y. Halpern. *An analysis of first order logics for reasoning about probability.* Artificial Intelligence., 1990. (Cited on page 19.)

[Hua 2008] Ming Hua, Jian Pei, Wenjie Zhang et Xuemin Lin. *Ranking queries on uncertain data: a probabilistic threshold approach*. In SIGMOD, 2008. (Cited on pages 22 et 23.)

[Ilyas 2004] Ihab F. Ilyas, Walid G. Aref et Ahmed K. Elmagarmid. *Supporting Top-k Join Queries in Relational Databases.* VLDB Journal, vol. 13, pages 207–221, 2004. (Cited on pages 24 et 25.)

[Ilyas 2008] Ihab F. Ilyas, George Beskales et Mohamed A. Soliman. *A survey of topk query processing techniques in relational database systems.* ACM Comput. Surv, vol. 40, 2008. (Cited on pages 23 et 25.)

[Imielinski 1984] Tomasz Imielinski. *Incomplete Information in Relational Databases.* ACM Journal., vol. 31, pages 761–791, 1984. (Cited on pages 18 et 20.)

[Jha 2010] Abhay Jha, Dan Olteanu et Dan Suciu. *Bridging the gap between intensional and extensional query evaluation in probabilistic databases*. In EDBT '10 Proceedings of the 13th International Conference on Extending Database Technology, pages 323–334, 2010. (Cited on page 73.)

[Jhingran 2006] Anant Jhingran. *Enterprise information mashups: integrating information, simply.* In VLDB, pages 3–4, 2006. (Cited on page 31.)

[Li 2009] Jian Li et Amol Deshpande. *Consensus answers for queries over probabilistic databases.* In PODS, 2009. (Cited on page 23.)

[linh Truong 2009] Hong linh Truong et Schahram Dustdar. *On Analyzing and Specifying Concerns for Data as a Service.* In The 2009 Asia-Pacific Services Computing Conference (IEEE APSCC 2009), pages 7–11, 2009. (Cited on pages 27 et 31.)

[Magnani 2010] Matteo Magnani et Danilo Montesi. *A Survey on Uncertainty Management in Data Integration.* J. Data and Information Quality, vol. 2, 2010. (Cited on page 26.)

[Marian 2011] Amelie Marian et Minji Wu. *Corroborating Information from Web Sources.* IEEE Data Engineering Bulletin, vol. 34, pages 11–17, 2011. (Cited on pages 17, 52, 56 et 57.)

[Olteanu 2009] Dan Olteanu, Jiewen Huang et Christoph Koch. *SPROUT: Lazy vs. eager query plans for tuple-independent probabilistic databases.* In 25th IEEE Int. Conf. on Data Eng, 2009. (Cited on page 28.)

[Papazoglou 2003] Mike P. Papazoglou. *Service-Oriented Computing: Concepts, Characteristics and Directions.* In 4th International Conference on Web Information Systems Engineering, WISE 2003, Rome, Italy, December 10-12, 2003, pages 3–12, 2003. (Cited on page 2.)

[Poole 2003] David Poole. *First order probabilistic inference.* In Proc. 18th Int. Joint Conference on Artificial Intelligence, page 985â991, 2003. (Cited on page 28.)

[Re 2006] Christopher Re, Nilesh N. Dalvi et Dan Suciu. *Query evaluation on probabilistic databases.* IEEE Data Eng. Bull, vol. 29, pages 25–31, 2006. (Cited on page 28.)

[Re 2007] Christopher Re et Dan Suciu. *Materialized views in probabilistic databases: for information exchange and query optimization*. In VLDB 07 Proceedings of the 33rd international conference on Very large data bases, pages 51–61, 2007. (Cited on page 19.)

[Sabesan 2009] Manivasakan Sabesan et Tore Risch. *Adaptive parallelization of queries over dependent web service calls*. In International conference on Data Engineering, pages 1725–1732, 2009. (Cited on page 74.)

[Sabesan 2012] Manivasakan Sabesan et Tore Risch. *Adaptive Parallelization of Queries to Data Web Service Operations.* Large-Scale Data-and Knowledge-Centered Systems, vol. 5, pages 49–69, 2012. (Cited on page 24.)

[Sadri 1991] Fariba Sadri. *Modeling Uncertainty in Databases*. In Proc. Seventh IEEE Int. Conf. Data Eng. (ICDE), 1991. (Cited on pages 18, 56, 57, 58 et 60.)

[Sadri 1995] Fereidoon Sadri. *Information source tracking method: Efficiency issues.* TKDE., vol. 7, pages 947–954, 1995. (Cited on pages 21 et 22.)

[Sen 2007] Prithviraj Sen et Deshpande A. *Representing and Querying Correlated Tuples in Probabilistic Databases*. In Data Engineering 2007. ICDE 2007. IEEE 23rd International Conference, pages 596–605, 2007. (Cited on page 28.)

[Sen 2008] Prithviraj Sen, Amol Deshpande et Lise Getoor. *Exploiting shared correlations in probabilistic databases*. In Proc.Very Large Data Bases, page 809â820, 2008. (Cited on page 28.)

[Soliman 2008] Mohamed A. Soliman, Ihab F. Ilyas et Kevin C. Chang. . *Probabilistic Top-k and Ranking-Aggregate Queries.* ACM Trans. Database Syst., vol. 33, 2008. (Cited on page 22.)

[Soliman 2009] Mohamed A. Soliman et Ihab F. Ilyas. *Ranking with uncertain scores*. In International Conference on Data Engineering ICDE, 2009. (Cited on pages 23 et 24.)

[Soliman 2010] Mohamed A. Soliman, Mina Saleeb et Ihab F. Ilyas. *Towards uncertainty-aware and rank-aware mashups*. In ICDE 2010, pages 1137–1140, 2010. (Cited on pages 17, 23, 27, 52 et 57.)

[Srivastava 2006] Utkarsh Srivastava, Kamesh Munagala, Jennifer Widom et Rajeev Motwani. *Query Optimization over Web Services*. In VLDB, pages 355–366, 2006. (Cited on pages 24, 60 et 74.)

[Stoyanovich 2011] Julia Stoyanovich, Susan B. Davidson, Tova Milo et Val Tannen. *Deriving Probabilistic Databases with Inference Ensembles*. In International Conference on Data Engineering ICDE, pages 303–314, 2011. (Cited on page 19.)

[Suciu 2011] Dan Suciu, Dan Olteanu, Christopher Re et Christoph Koch. Probabilistic databases. Morgan and Claypool Publishers, 2011. (Cited on page 65.)

[Tabatabaei 2008] Sayed Gholam Hassan Tabatabaei, Wan M. N. Wan-Kadir et Suhaimi Ibrahim. *A Comparative Evaluation of State-of-the-Art Approaches for Web Service Composition*. ICSEA, vol. 4, pages 488–493, 2008. (Cited on page 13.)

[Tatbul 2004] Nesime Tatbul, Mark Buller, Reed Hoyt, Steve Mullen et Stan Zdonik. *Confidence-based data management for personal area sensor networks*. In DMSN '04 Proceeedings of the 1st international workshop on Data management for sensor networks:in conjunction with VLDB 2004, pages 24–31, 2004. (Cited on pages 17, 27 et 53.)

[Tatemura 2008] Junichi Tatemura, Songting Chen, Fenglin Liao, Oliver Po, K. Selcuk Candan et Divyakant Agrawal. *UQBE: uncertain query by example for web service mashup*. In SIGMOD Conference, pages 1275–1280, 2008. (Cited on page 25.)

[Vaughan-Nichols 2002] Steven J. Vaughan-Nichols. *Web Services: Beyond the Hype*. IEEE Internet Computing, vol. 35, no. 2, pages 18–21, 2002. (Cited on page 11.)

[Weise 2008] Thomas Weise, Steffen Bleul, Diana Comes et Kurt Geihs. *Different Approaches to Semantic Web Service Composition*. In Third International Conference on Internet and Web Applications and Services, pages 90–96, 2008. (Cited on page 13.)

[Widom 2005] Jennifer Widom. *Trio: A System for Integrated Management of Data, Accuracy, and Lineage*. In Proceedings of the Second Biennial CIDR, pages 262–276, 2005. (Cited on page 18.)

[Widom 2008] Jennifer Widom. *Trio: a system for data, uncertainty, and lineage.* Charu Aggarwal, editor, Managing and Mining Uncertain Data, chapter 5. Springer-Verlag, 2008, 2008. (Cited on page 17.)

[Wu 2009] Jian Wu, Qianhui Liang et Hengyi Jian. *Bayesian network based services recommendation. Services.* In Computing Conference, 2009. APSCC 2009. IEEE Asia-Pacific Singapore, pages 313–318, 2009. (Cited on page 27.)

[Yu 2008] Qi Yu, Xumin Liu, Athman Bouguettaya et Brahim Medjahed. *Deploying and managing Web services: issues, solutions, and directions*. VLDB Journal, vol. 17, no. 3, pages 86–93, 2008. (Cited on pages 13, 14, 36, 52 et 63.)

[Zadeh 1965] Lotfi Zadeh. *Fuzzy sets.* Information and Control, vol. 8, pages 338–353, 1965. (Cited on pages 15 et 34.)

[Zhao 2008] Zhuofeng Zhao, Jun Fang et Jing Cheng. *CAFISE-S: An Approach to Deploying SOA in Scientific Information Integration.* In ICWS '08. IEEE International Conference on Web Services., pages 425–432, 2008. (Cited on page 31.)