Université Djillali Liabes de Sidi Bel Abbes
Evolutionary Engineering & Distributed Information Systems Laboratory

Université Claude Bernard Lyon 1
Laboratoire d'InfoRmatique en Image et Systèmes d'information
Ecole Doctorale Informatique et Mathématiques de Lyon

Numéro d'ordre: **-****

Année 2015

Thèse en cotutelle

Entre

L'Université Djillali Liabes de Sidi Bel Abbes, Algérie

Et

L'Université Claude Bernard Lyon 1, France

En vue d'obtenir le titre de

Doctorat $3^{éme}$ cycle LMD
Spécialité: informatique

Docteur de l'université Lyon 1
Spécialité: informatique

Présentée par

MALKI Abdelhamid

Modélisation sémantique du cloud computing : vers une composition de services DaaS à sémantique incertaine

Thèse soutenue le 23 avril 2015 devant le jury composé de:

Rapporteurs:	BELALEM Ghalem	–	Professeur à l'Université d'Oran 1, Algérie
	BELLATRECHE Ladjel	–	Professeur à l'ISAE-ENSMA de Poitiers, France
Examinateurs:	BENMOHAMMED Mohamed	–	Professeur à l'Université Constantine 2, Algérie
	AIT AMEUR Yamine	–	Professeur à l'INPT-ENSEEIHT de Toulouse France
Directeur:	BENSLIMANE Djamal	–	Professeur à l'Université Lyon 1, France
Directeur:	BENSLIMANE Sidi Mohamed	–	Professeur à l'Université de Sidi Bel Abbes, Algérie

Dédicace

Louange à Dieu, Seigneur des mondes ;
C'est toi que nous adorons et de toi que nous implorons secours ;
Que la paix et la bénédiction soit sur son dernier messager ;
A mes très chers parents,
A mes frères,
A toute ma famille,
A mes amis.

Remerciements

On passe tous par des moments de doute, d'incertitude, de petites et grandes crises ... et on ne peut les dépasser que par le soutien de tous ceux qui nous relèvent de nos petites peines. Pour cela et pour bien d'autres raisons je tiens à remercier :

Monsieur **Belalem Ghalem**, Professeur à l'université d'Oran, et Monsieur **BELLATRECHE Ladjel**, Professeur à l'ISAE-ENSMA de Poitiers, qui m'ont fait l'honneur de rapporter mon travail et pour leurs remarques enrichissantes.

Monsieur **BENMOHAMMED Mohamed**, Professeur à l'université de Constantine 2, et Monsieur **AIT AMEUR Yamine**, Professeur à l'INPT-ENSEEIHT de Toulouse, qui ont accepté de faire partie de mon jury en tant qu'examinateur.

Mon directeur de thèse, **Benslimane Sidi Mohamed** pour sa sympathie, son énergie positive, pour les discussions de travail, sa disponibilité et son écoute.

Mon directeur de thèse, **Benslimane Djamal** de m'avoir accueillie dans son équipe, pour tous ses conseils, ses critiques, ses encouragements dont j'ai bénéficié tout au long de ce travail, et pour m'avoir témoigné sa confiance totale.

Monsieur **barhamgi mahmoud** pour l'intérêt qu'il a porté à mes questions. Les discussions échangées tout au long de ma thèse m'ont permis de bénéficier de son expertise, réflexion et critique.

Mes parents pour leur patience et leur soutien indéfectible qui ont été plus qu'indispensables.

Résumé

Avec l'émergence du mouvement Open Data, des centaines de milliers de sources de données provenant de divers domaines (e.g., santé, gouvernementale, statistique, etc.) sont maintenant disponibles sur Internet. Ces sources de données sont accessibles et interrogées via des services cloud DaaS, et cela afin de bénéficier de la flexibilité, l'interopérabilité et la scalabilité que les paradigmes SOA et Cloud Computing peuvent apporter à l'intégration des données. Dans ce contexte, les requêtes sont résolues par la composition de plusieurs services DaaS. Définir la sémantique des services cloud DaaS est la première étape vers l'automatisation de leur composition. Une approche intéressante pour définir la sémantique des services DaaS est de les décrire comme étant des vues sémantiques à travers une ontologie de domaine. Cependant, la définition de ces vues sémantiques ne peut pas être toujours faite avec certitude, surtout lorsque les données retournées par un service sont trop complexes. Dans cette thèse, nous proposons une approche probabiliste pour représenter les services DaaS à sémantique incertaine. Dans notre approche, un service DaaS dont la sémantique est incertaine est décrit par plusieurs vues sémantiques possibles, chacune avec une probabilité. Les services ainsi que leurs vues sémantiques possibles sont représentées dans un registre de services probabiliste (PSR). Selon les dépendances qui existent entre les services, les corrélations dans PSR peuvent être représentées par deux modèles différents : le modèle Bloc-indépendant-disjoint (BID), et le modèle à base des réseaux bayésiens. En se basant sur nos modèles probabilistes, nous étudions le problème de l'interprétation d'une composition existante impliquant des services à sémantique incertaine. Nous étudions aussi le problème de la réécriture de requêtes à travers les services DaaS incertains, et nous proposons des algorithmes efficaces permettant de calculer les différentes compositions possibles ainsi que leurs probabilités. Nous menons une série d'expérimentation pour évaluer la

performance de nos différents algorithmes de composition. Les résultats obtenus montrent l'efficacité et la scalabilité de nos solutions proposées.

Abstract

With the emergence of the Open Data movement, hundreds of thousands of datasets from various concerns (e.g., healthcare, governmental, statistic, etc.) are now freely available on Internet. A good portion of these datasets are accessed and queried through Cloud DaaS services to benefit from the flexibility, the interoperability and the scalability that the SOA and Cloud Computing paradigms bring to data integration. In this context, user's queries often require the composition of multiple Cloud DaaS services to be answered. Defining the semantics of DaaS services is the first step towards automating their composition. An interesting approach to define the semantics of DaaS services is by describing them as semantic views over a domain ontology. However, defining such semantic views cannot always be done with certainty, especially when the service's returned data are too complex. In this dissertation, we propose a probabilistic approach to model the semantic uncertainty of data services. In our approach, a DaaS service with an uncertain semantics is described by several possible semantic views, each one is associated with a probability. Services along with their possible semantic views are represented in probabilistic service registry (PSR).According to the services dependencies, the correlations in PSR can be represented by two different models :Block-Independent-Disjoint model (noted BID), and directed probabilistic graphical model (Bayesian network). Based on our modeling, we study the problem of interpreting an existing composition involving services with uncertain semantics. We also study the problem of compositing uncertain DaaS services to answer a user query, and propose efficient methods to compute the different possible compositions and their probabilities. We conduct a series of experiments to evaluate the performance of our composition algorithms. The obtained results show the efficiency and the scalability of our proposed solutions.

Table des matières

Table des figures

Liste des tableaux

Chapitre 1

Introduction Générale

1 Contexte

Au cours des dernières années, le mouvement des données ouvertes (en anglais, *open data*) a été largement adopté par les gouvernements et les organisations [54]. Ceci est motivé par la grande transparence et l'ouverture au monde que les corps gouvernementaux et les entreprises peuvent acquérir en publiant certaines de leurs données via le Web. Un nombre important de portails open data (e.g., data.gov, data.gov.uk, Data.gouv.fr, etc.) ont été récemment conçu et proposé pour la publication de centaines de milliers de sources de données qui peuvent être accédées et réutilisées par les entreprises et les citoyens. Les sources de données publiées concernent en premier lieu les administrations et les organisations publiques qui ouvrent les données qu'elles produisent dans le cadre de leurs activités quotidiennes (gouvernementales, commerciales, industrielles, santé, scientifiques, statistiques, etc.).

Cependant, la plupart des ensembles de données publiées sont décrites sous des formats et des descriptions hétérogènes, voire brutes (i.e., non-structurée) [71], ce qui cause des problèmes d'interopérabilité empêchant leur réutilisation et leur intégration avec d'autres sources de données. Actuellement, plusieurs approches [30, 71, 102], architectures [15], et protocoles (e.g., Socrata [1], OData [2]) ont été proposés dans le but de rendre les données ouvertes interprétables (lisibles par les machines), et interopérables. La plupart de ces

1. http ://opendata.socrata.com.
2. http ://www.odata.org

travaux reposent sur l'utilisation des APIs Web (e.g., services Web) qui fournissent une interface standardisée pour les open data. Un service web est une entité logicielle disposant d'une interface qui peut être définie, décrite, et découverte via un langage universel (i.e., XML). La classe des services Web destinée pour le traitement et la gestion des données est appelée service *DaaS* (Data as a Service) ou *service fournisseur de données* [18]. Cette catégorie de services est motivée par la flexibilité et l'interopérabilité que l'architecture orientée service (SOA) pourrait apporter à l'intégration des données. En outre, les services DaaS sont une des catégories de services cloud destinés pour le traitement et la gestion des données dans les environnements cloud computing. Actuellement, Le cloud computing est la solution la plus favorable pour les entreprises et les organisations notamment celles qui ont un budget informatique très réduit, afin de bâtir une infrastructure physique puissante (traitement, stockage, etc), scalable, auto-reconfigurable et sans avoir besoin d'un investissement initial important. Dans ce contexte, le cloud computing est considéré comme un support idéal pour le déploiement et l'hébergement des données ouvertes qui se caractérisent par une taille très étendue.

Dans les systèmes d'intégration de données à base de services, les requêtes utilisateurs sont exprimées de façon déclarative (e.g., SQL, SPARQL, etc), et résolues en composant un ensemble de services, i.e., sélectionner et combiner un ensemble de services pertinents de telle façon que leur union couvre tous les besoins de la requête utilisateur. Durant ces dernières années, d'importants travaux de recherche ont été menés sur la découverte et la composition des services DaaS [8, 9, 101, 120, 124, 125]. La plupart de ces travaux exploitent la sémantique des services DaaS pour automatiser entièrement le processus de composition.

Les services dans les différentes couches du cloud computing peuvent être sémantisés en décrivant leurs capacités et leurs interprétations à travers une connaissance partagée (i.e., ontologie), ce que nous entendons par la *Modélisation Sémantique du Cloud Computing* [110, 111]. Cette sémantisation permet d'automatiser les tâches liées à l'utilisation des services cloud, par exemple : sélectionner le service IaaS qui fournit la machine virtuelle la plus puissante ; composer deux services SaaS ou DaaS tout en répondant à une requête complexe, etc. Plusieurs modèles sémantiques peuvent être utilisés pour sémantiser les services cloud, e.g., OWL-S [7], WSMO [100], SAWSDL [65], etc. Toutefois, le modèle à

base de vues sémantiques [9,120] (i.e., sémantique déclarative) est le plus adéquat pour les services DaaS. Ceci s'explique par le fait que l'interprétation d'un service DaaS dépend non seulement de ses entrées/sorties, mais aussi de la façon dont il accède aux données. Dans ce travail, nous adoptons l'approche proposée dans [9], où la sémantique d'un service DaaS est modélisée comme étant une *vue RDF* (i.e., vue sémantique) à travers une ontologie de domaine.

2 Problématique

Généralement, un service DaaS est associé à une et une seule vue sémantique. Cela se produit lorsque le processus de sémantisation est réalisé manuellement par des annotateurs, i.e., les personnes chargées de la définition des vues sémantiques sont en mesure d'interpréter correctement les données encapsulées par un service DaaS, et peuvent par conséquent définir avec certitude et exactitude la vue sémantique correspondante.

Cependant, dans certains cas, la sémantique des données encapsulées par un service est *incertaine* et peut donc accepter différentes interprétations possibles. Le service doit ensuite être associé à plusieurs vues sémantiques (probables) représentant ses différentes interprétations possibles. Cela peut se produire pour différentes raisons. Par exemple, l'annotateur et le fournisseur de service peuvent être deux entités indépendantes, où l'annotateur n'a pas suffisamment d'informations sur le service publié (e.g., Meta-data, description textuelle, etc), ou lorsque le fournisseur lui-même n'a pas des connaissances approfondies sur les données accédées par son service. Ce dernier cas est particulièrement courant dans le contexte des données ouvertes où souvent la sémantique des données publiées est imprécise voire inexacte.

En outre, la sémantisation manuelle des services est souvent coûteuse en termes de temps, ce qui la rend impraticable. Ceci est dû à la prolifération des données ainsi qu'à leur diversité. Par exemple, le portail Open Data (opendata.socrata.com) contient presque 23000 sources données qui appartiennent à différents domaines (e.g., gouvernemental, biomédical, économique, éducatif, etc). Une des solutions qui peut être adoptée pour combler ces carences est d'automatiser le processus de sémantisation des services. Cette automatisation se base principalement sur les techniques de *Schema Matching* [69, 106].

Ces dernières génèrent la sémantique d'un service en comparant ses capacités (i.e., entrées, sorties, données encapsulées, description textuelle, etc) avec les éléments d'une ontologie de domaine (e.g., Class, ObjetProperty, DataProperty, etc). Cependant, la plupart des techniques de matching utilisent des mesures de similarité approximatives et incertaines, ce qui permet d'associer au même service plusieurs vues sémantiques *incertaines*, chacune avec un poids qui définit son degré de certitude (i.e., probabilité).

En plus de l'incertitude, les services peuvent être soit *indépendants*, soit *dépendants*. Pour le premier cas, les sources de données encapsulées par les services sont différentes, i.e., une source de données est accédée par un et un seul service. Dans une telle situation, les vues sémantiques possibles qui sont associées à des services différents sont considérées comme étant des évènements indépendants. Par contre, lorsque les services sont dépendants (i.e., ils accèdent aux mêmes sources de données) différentes corrélations peuvent exister entre l'ensemble des vues sémantiques possibles de tous les services. Ces corrélations permettent à une source de données d'être interprétée de façon cohérente et non-ambiguë. Ceci est particulièrement fréquent lorsque deux ou plusieurs services qui partagent la même source de données sont impliqués dans la même composition. Dans une telle situation, Les vues sémantiques associées aux services impliqués doivent interpréter de façon cohérente les données communes.

En résumé, l'objectif de cette thèse est de proposer une approche pour la gestion de l'incertitude liée à l'interprétation des services DaaS. La gestion de l'incertitude comporte plusieurs aspects, à savoir, la représentation des services incertains, l'évaluation de requêtes en présence de services incertains, la corrélation entre les services, etc. Dans la section suivante, nous présentons de façon détaillée les principaux challenges posés dans le cadre de cette thèse.

3 Challenges

Dans cette section, nous résumons les principaux challenges qui doivent être traités lors de l'utilisation des services incertains.

– *Représentation des services à sémantique incertaine* : Lorsque la sémantique des services est incertaine, un service est décrit par plusieurs vues sémantiques. Ces

vues sémantiques décrivent les différentes interprétations possibles des données qu'un service DaaS peut retourner. Chaque vue sémantique possible a un poids qui définit son degré de certitude. Les principales questions qui se posent alors sont : Comment calculer le degré de certitude des vues sémantiques ? ; Comment le quantifier (modèle possibiliste, probabiliste, etc) ? ; et comment représenter et interpréter l'ensemble de services avec leurs vues sémantiques possibles ?.

– *Représentation des services à sémantique corrélée* : Lorsque les services DaaS encapsulent les mêmes sources de données (i.e., services dépendants), il est nécessaire de maintenir les dépendances qui existent entre leurs vues sémantiques possibles. Il est alors important de reconsidérer les questions précédentes pour tenir compte de la dimension de la corrélation. Ainsi, il est question d'étudier comment modéliser les corrélations qui peuvent exister entre les différents services incertains et de propager cette corrélation dans le calcul de la composition de services.

– *Traitement de requêtes à travers les services incertains* : Le traitement de requêtes dans un contexte orienté services consiste à composer des services qui couvrent la requête. En présence de services incertains, la composition de services ne peut être qu'incertaine, c'est-à-dire que la même composition répondra en partie aux besoins de la requête et fournira aussi une ou plusieurs autres sémantiques qui ne correspondent pas à la requête initiale. Il est donc important de revisiter le traitement de requêtes en présence de services à sémantiques incertaines d'abord dans le cas non corrélé, puis dans le cas corrélé pour la prise en compte de la préservation de la corrélation dans le processus de réécriture de requêtes.

4 Contributions

Pour traiter les challenges ci-dessus, nous proposons une approche probabiliste permettant la représentation et la composition des services DaaS incertains [74, 75]. Nos principales contributions sont résumées comme suit :

– *Concept de services à sémantiques probabilistes* : nous proposons un modèle probabiliste pour les services à sémantiques incertaines. Ce modèle associe aux différentes

vues sémantiques possibles d'un même service des probabilités qui qualifient leurs degrés de certitude. Les services et leurs vues sémantiques possibles sont représentés dans un registre de services probabilistes (noté PSR) qui suit le modèle BID (*Bloc-Indépendant-Disjoint*), de telle sorte que d'une part les interprétations possibles d'un même service sont disjointes, et d'autre part les interprétations associées à des services différents sont indépendantes entre elles. Nous proposons aussi une définition formelle de l'interprétation probabiliste d'une composition de services à sémantiques psrobabilistes.

– *Représentation bayésienne du registre de services probabiliste* : nous proposons de représenter les corrélations entre les vues sémantiques probabilistes des services d'un registre PSR de façon générique en utilisant les réseaux bayésiens (i.e., graphe probabiliste orienté). Cette représentation est réalisée en introduisant des variables aléatoires, et des distributions de probabilités conditionnelles entre elles.

– *Traitement de requêtes par composition de services probabilistes* : nous proposons une approche de génération automatique de compositions de services à sémantiques incertaines pour les besoins de traitement de requêtes. Cette approche est définie pour le cas de services non corrélés et de services corrélés. Traiter une requête utilisateur Q à travers un registre de services probabilistes revient à évaluer la requête sur chacun des registres de services possibles issus de la théorie des mondes possibles. Un registre possible de services est un registre déterministe dans le sens où chaque service possède une et une seule sémantique. En revanche, le registre lui même n'est que probable. Le nombre de registres de services possibles peut être extrêmement large, et par conséquent l'évaluation des requêtes devient difficile voire impossible. Dans ce sens, nous proposons deux algorithmes (l'un pour le PSR à base de BID, et l'autre pour le PSR bayésien) permettant d'évaluer efficacement les requêtes utilisateurs (i.e., dans un temps raisonnable), et cela sans avoir besoin de matérialiser l'ensemble des registres de services possibles. Nous proposons aussi un algorithme qui implémente une stratégie permettant de générer efficacement l'ensemble des $Top\text{-}k$ compositions.

– *Implémentation et évaluation* : Les concepts proposés sur un plan théorique sont complétés par des implémentations. Ainsi, les représentations des services incertains sont réalisées. Il en est de même pour l'algorithme de génération de compositions pour le traitement de requêtes. Nous proposons aussi une expérimentation de notre prototype et une évaluation de la compléxiité de nos algorithmes.

5 Organisation de la thèse

Le reste de la thèse est organisé comme suit :

Dans le chapitre 2, nous présentons les notions de base des concepts que nous jugeons nécessaires à la compréhension du contenu de cette thèse. Nous présentons d'abord les concepts de base de l'architecture orientée services. Ensuite, nous présentons les notions fondamentales des bases de données probabilistes, leur interprétation, ainsi que la sémantique des requêtes incertaines. Enfin, nous terminons ce chapitre par quelques notions de base sur les réseaux bayésiens.

Dans le chapitre 3, nous présentons notre modèle de *Registre de services probabilistes* (PSR) qui permet de représenter la sémantique incertaine des services DaaS. Dans cette partie nous supposons que les services sont indépendants (i.e., ils n'encapsulent pas les mêmes sources de données), par conséquent, les dépendances probabilistes entre les vues sémantiques sont définies par le modèle BID. En se basant sur le concept PSR, nous décrivons l'interprétation d'une composition incertaine, ainsi que la sémantique de l'évaluation de requêtes. Ensuite, nous proposons un théorème qui nous permet de calculer les probabilités des compositions dans un temps polynomial, et par conséquent assurer l'évaluation efficace des requêtes. Un algorithme de $Top\text{-}k$ est également proposé. Aussi dans ce chapitre, nous définissons les différents types de la sémantique incertaine, ainsi qu'une méthode pour la génération des sémantiques probabilistes.

Dans le chapitre 4, nous adressons le problème de la corrélation entre les sémantiques incertaines. Dans un premier temps, nous décrivons la représentation bayésienne d'un registre de services probabilistes. Ensuite, nous définissons la sémantique de l'évaluation des requêtes à travers un PSR corrélé. Une approche pour l'évaluation efficace des requêtes est également présentée. Cette approche se base sur l'algorithme d'inférence *Variable Eli-*

mination pour calculer les probabilités des compositions obtenues.

Dans le chapitre 5, nous présentons les systèmes et les architectures implémentant nos différentes contributions (incertitude avec corrélation et sans corrélation). Ensuite, nous analysons la complexité des différents algorithmes proposés dans le cadre de cette thèse. La dernière partie de ce chapitre est consacrée aux expérimentations que nous avons réalisées et aux résultats obtenus.

Le chapitre 6 dresse un état de l'art des travaux qui s'apparentent à notre problématique. L'objectif de cette partie est de positionner notre travail par rapport aux travaux existants.

Le chapitre 7 synthétise ce mémoire par une conclusion qui discute les contributions réalisées dans le cadre de nos travaux de thèse, ainsi que les perspectives de recherche envisagées.

Chapitre 2

Background

Dans ce chapitre, nous commençons par décrire les concepts de base de l'architecture orientée service. Ensuite, nous présentons les bases de données probabilistes, leur interprétation, ainsi que la sémantique des requêtes. Enfin, nous concluons ce chapitre par quelques notions de base sur les réseaux bayésiens.

1 Architecture orientée service

De nos jours, la plupart des systèmes d'information sont devenus de plus en plus distribués, complexes et coûteux en termes de gestion. Cette évolution du monde informatique a entrainée le développement de nouvelles architectures et technologies qui facilitent la création et le déploiement des applications. En effet, l'apparition de l'approche orientée objet [116] a donné lieu à de nouvelles technologies de distribution des applications [37] (i.e., middleware orienté objet) telles que RMI (Remote Method Invocation) et CORBA (Common Object Request Broker Architecture). Les principes du modèle orienté objet ont été étendus et améliorés par une autre approche conçue aussi pour la conceptualisation des systèmes distribués, appelée approche orientée composant [115] (i.e., middleware orienté composant). Un composant est une boite noire regroupant un ensemble d'objets interdépendants (i.e., homogènes en terme de fonctionnalité), et interagissant avec l'extérieur à travers une interface dédiée. Cependant, l'approche orientée composant est freinée par plusieurs obstacles, à savoir l'hétérogénéité techniques des composants (i.e., hétérogénéité des langages de programmation : JEE, .NET ; hétérogénéité des protocoles de commu-

nication : IIOP, RMI ; etc) ce qui empêche les différents types d'applications telles que les applications web (e.g. ; PHP, ASP, JSP, etc), et les applications mobiles (e.g., ; Android, IOS, etc) d'interagir avec ces composants logiciels. Ainsi, la forte dépendance (i.e., moins d'autonomie) qui existe entre les composants au-sein d'une application empêche leur évolution et leur maintenabilité. Ces lacunes font naître le besoin d'une architecture plus flexible et plus agile où les applications sont totalement indépendantes autonomes et facilement intégrables. D'où l'apparition de l'architecture orientée service (en anglais, Service-Oriented Architecture (SOA)) [113].

1.1 Principes de la SOA

La généralisation et la démocratisation de l'accès à l'internet, ainsi que le besoin d'interopérabilité ont entrainé la venue d'un nouveau paradigme qui présente un certain nombre d'avantages pour l'interaction entre les applications, appelé Architecture Orientée-Service. L'architecture orientée service est une approche architecturale qui consiste à créer et développer des applications comme étant des services fournis et réutilisés à la demande [91]. Autrement dit, la SOA permet de concevoir un système d'information en se basant sur un ensemble de services développés dans différents langages de programmation, déployés sur différentes plates-formes avec divers modèles de sécurité et processus métier [10].

D'après les définitions précédentes, l'architecture orientée service est basée principalement sur la notion de service. Un service est matérialisé par un composant logiciel fournissant une ou plusieurs fonctionnalités accessibles via son interface. L'interface du service est une description universelle qui définit et expose aux applications clientes les informations nécessaires à sa réutilisation (i.e., son invocation).

L'utilisation de ces services au sein d'un système d'information offre de nombreux avantages, à savoir : le *couplage faible*, l'*interopérabilité*, et la *transparence*. Le couplage faible signifie que les services peuvent être mis en interaction, tout en restant indépendants les uns des autres, et sans dévoiler leur implémentation interne. L'interface universelle des services permet de combler les conflits d'interopérabilité qui peuvent exister entre les entités hétérogènes. La transparence permet à un service d'être utilisé sans avoir besoin des connaissances liées à son implémentation ainsi qu'à sa localisation.

L'architecture orientée service repose sur un modèle qui décrit un ensemble d'interactions entre trois types d'acteurs : ***fournisseur de service***, ***Client de service***, ***registre de services*** [89]. Les interactions entre ces trois acteurs peuvent être résumées en trois primitives de communication : la ***publication***, la ***découverte***, et l'***invocation***. Les services sont créés est déployés par les fournisseurs. Les fournisseurs de service enregistrent la description de leurs services dans le registre (i.e. ; étape1 : la publication). Les consommateurs de service envoient des requêtes au registre pour découvrir un service qui répond à leurs besoins (i.e. ; étape2 : la découverte). Une fois le service choisi, le client de service peut se connecter au fournisseur et utiliser le service à travers sa description (i.e. ; étape3 : l'invocation). Ce modèle d'interaction est illustré dans la figure 2.1.

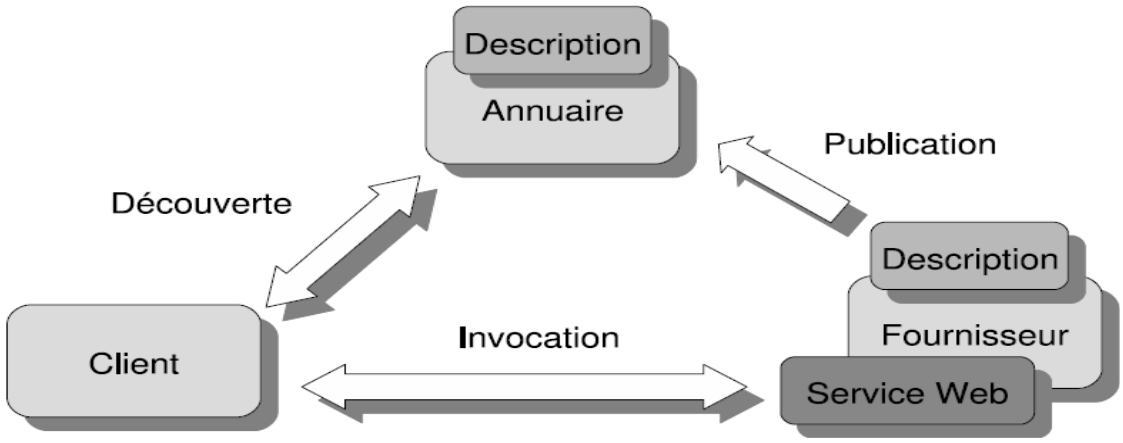


FIGURE 2.1 – Interactions au sein d'une architecture orientée service.

L'objectif de l'architecture orientée service est de fournir un accès simple et rapide aux fonctionnalités fournies par des tiers distribués et hétérogènes. En ce sens, il faut mettre en place un environnement qui décrit les modalités d'interaction et d'utilisation des services. Cet environnement inclue un ensemble de standards, protocoles, et mécanismes qui peuvent être regroupés en deux catégories : mécanismes fonctionnels et mécanismes non-fonctionnels. Les mécanismes fonctionnels décrivent les principales tâches liées à l'utilisation des services à savoir, la publication, la découverte, la composition, et l'invocation. Les mécanismes non-fonctionnels prennent en charge les besoins optionnels notamment la sécurité, les transactions, et la qualité de service.

1.2 Service web

Les services Web sont certainement la technologie la plus populaire dans le monde industriel et académique pour migrer vers l'architectures à services. Les principaux mécanismes de fonctionnalités requises pour la réalisation d'une SOA sont assurés par une pile de standard de protocoles construite autour de la définition des services web. En fait, il existe deux types de services web, les services web SOAP, et les services web REST. Ces deux types de services web proposent différents standards et langages afin de mettre en place les principes de la SOA.

Service web SOAP

Les services web SOAP sont la première technologie qui a été proposée dans le but d'implémenter l'architecture SOA. Cette réalisation de la SOA est assurée par une panoplie de standards qui sont tous basés sur le modèle XML. XML est un format universel compréhensible par toutes les plateformes de programmation ; ce qui respecte les principes de la SOA, tels que : le couplage faible, et l'interopérabilité. Le W3C définit les services web SOAP comme étant [121] :
≪ *Un service web est une entité logicielle capable d'être interopérable avec d'autres systèmes distribués. Il a une interface décrite dans une description facilement exploitable, appelée WSDL. Les autres systèmes interagissent avec le service web en se basant sur des messages SOAP, généralement transmis via le protocole HTTP avec une sérialisation XML.* ≫

Autour de cette définition, les services web SOAP reposent sur un ensemble de standards décrits comme suit :

- **SOAP** [1] : Simple Object Access Protocol (SOAP) est un protocole dont la syntaxe est basée sur XML. Il permet des échanges standardisés de données, tout en résolvant les conflits syntaxiques et techniques. Il permet d'invoquer des services indépendamment de la plate-forme d'exécution. En outre, il s'appuie sur d'autres protocoles de la couche d'application, notamment HTTP, et cela pour la négociation et la transmission des messages.
- **WSDL** [2] : Web Services Description Language (WSDL) est un langage de descrip-

1. http ://www.w3.org/TR/soap/
2. http ://www.w3.org/TR/wsdl

tion à base d'XML utilisé pour décrire les fonctionnalités offertes par un service web (i.e. ; l'interface du service). Le fichier WSDL d'un service définit la façon dont le service peut être appelé, avec quels paramètres, et quelle est la structure des données qu'il retourne.

- **UDDI** [3] : Universal Description, Discovery and Integration (UDDI) est une plateforme indépendante, à base d'XML, qui assure le regroupement, le stockage et la diffusion des descriptions de services Web.
- **WS-BPEL** [4] : Web Services Business Process Execution Language (WS-BPEL) est une spécification du consortium OASIS. WS-BPEL est un langage d'orchestration de services basé sur XML, tout comme les autres standards. WS-BPEL permet de construire des compositions de services interprétables et exécutables par un moteur d'orchestration (i.e. ActiveBpel, OAEapache, etc).
- **WS-*** : Il existe une variété de spécifications associées aux Services Web WS-*. Ces spécifications sont à des niveaux de maturité parfois différents, et sont maintenus par diverses organisations de standardisation : WS-Security [5], WS-Trust [6], WS-Inspection, etc.

Service web REST

REST a été proposé par Roy Thomas Fielding dans sa thèse ≪*Architectural Styles and the Design of Network-based Software Architectures*≫ [40]. REST est l'acronyme de Representational State Transfer. C'est un style d'architecture qui se base sur un ensemble de contraintes qui permettent, lorsqu'elles sont appliquées aux composants d'une architecture, d'optimiser certains critères propres au cahier des charges du système à concevoir. Cela signifie que REST définit un ensemble de contraintes sur les systèmes hypermedia distribués comme le World Wide Web afin d'optimiser les qualités désirées comme la séparation des tâches, la simplicité, la généricité ou les performances réseaux. REST fournit un véritable modèle pour décrire le fonctionnement que devrait avoir le web aujourd'hui. Bien que moins formel, ce modèle peut être comparé aux paradigmes objet ou entité-association.

3. http ://www.oasis-open.org/committees/uddi-spec/doc/spec/v3/uddi-v3.0.2-20041019.htm
4. http ://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html
5. http ://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf
6. http ://docs.oasis-open.org/ws-sx/ ws-trust/200512/ws-trust-1.3-os.html

L'architecture REST suit les principes de base du modèle client/serveur. L'information de base, dans une architecture REST, est appelée *ressource*. Toute information qui peut être nommée est une ressource : un article d'un journal, une photo, un service ou n'importe quel concept. Chaque ressource est un composant distribué qui est géré par une norme, et une interface commune rendant possible la manipulation des ressources. En plus de la ressource, l'architecture REST est basée sur les éléments suivants :

- *Identifiant de la ressource* : Une ressource est identifiée par un identificateur de ressource. Il permet aux composants de l'architecture d'identifier les ressources qu'ils manipulent. Sur le web ces identificateurs sont les URI (Uniform Resource Identifier).
- *Représentation de la ressource* : Les composants de l'architecture manipulent ces ressources en transférant des représentations de ces ressources. Sur le web, on trouve aujourd'hui le plus souvent des représentations au format HTML, JSON ou XML.
- *Opération sur la ressource* : Les ressources sont manipulés par le transfert des représentations à travers une interface uniforme adressée par l'identifiant de ressource.

Chaque représentation doit mettre en œuvre une interface **CRUD** pour réaliser l'interface uniforme requise par l'architecture REST. Le protocole HTTP/1.1 définit huit méthodes, dont quatre permettent de définir l'interface : GET, PUT, POST et DELETE. Le tableau 2.1 montre les correspondances entre les méthodes HTTP et les actions **CRUD**. Toute représentation doit accepter toutes ces quatre méthodes, bien que cela ne signifie pas qu'ils doivent être mises en œuvre.

TABLE 2.1 – Correspondances entre les méthodes HTTP et les actions CRUD

Méthode	CRUD	Description
POST	Create	créer une nouvelle ressource
GET	Retrieve	Récupérer une ressource
PUT	Update	Mettre à jour une ressource
DELETE	Delete	Supprimer une ressource

Les services web REST sont apparus pour simplifier le développement, la publication et l'utilisation de services Web, tout en suivant les principes de l'architecture REST [92]. Ces services répondent aux requêtes des utilisateurs en fournissant des objets web (i.e. ressources) à travers les méthodes HTTP : Get, Head, Post, Put, Delete, Option.

L'une des raisons pour lesquelles la description de service constitue un élément important dans l'approche SOAP, est que chaque service expose une interface spécifique qui décrit ses fonctionnalités, et sans la description de cette interface, il est impossible d'utiliser ou d'invoquer ce service. En revanche, les services REST se basent sur un ensemble de contraintes qui permettent d'éliminer la nécessité d'avoir une description spécifique pour l'interface de service. De plus, la sélection des services REST est faite simplement en suivant des liens URI.

Cependant, Dans la pratique, la description des services REST est généralement utile, même si elle n'est pas strictement nécessaire. Par exemple, la description d'un service peut être utilisée comme une documentation de celui-ci, afin que les utilisateurs de ce service sachent à quoi s'attendre. Plusieurs langages ont été proposés pour décrire les services REST. WADL (*Web Application Description Language*) [49] est un langage conçu pour la description des services REST. L'avantage de WADL est qu'il fournisse une description à base d'XML rendant les services REST compréhensibles et interprétables par la machine. WADL décrit les ressources fournies par un service REST, ainsi que les relations qui existent entre eux sous la forme de liens. Un service est décrit en utilisant un ensemble de balises de type **Resource**. Chaque élément **Resource** décrit une ressource fournie par un service, ainsi que les méthodes appliquées sur celle-ci. Chaque méthode a des entrées (**Request**) et des sorties (**Response**). L'élément **Request** spécifie comment les entrées sont représentées, leurs types de données, etc. L'élément **Response** décrit la représentation des données retournées (e.g., HTML, JSON, CSV, XML, RDF, etc), et/ou des post-conditions. La figure 2.2 donne un exemple d'un service REST décrit avec le langage de description WADL.

Dans la littérature, il existe d'autres langages qui peuvent être utilisés pour décrire les services REST à savoir : WSDL (i.e., la description des services SOAP), hRESTS (HTML for RESTful Services) [64], ReLL(Resource Linking Language) [4], etc ;

1.3 Service web d'accès aux données

Ces dernières années ont vu un intérêt croissant pour l'utilisation des services web comme un support fiable pour l'édition et le partage des données entre les organisations. Les entreprises modernes sont orientées vers une architecture orientée services pour le

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<application xmlns="http://wadl.dev.java.net/2009/02">
   <doc xmlns:jersey="http://jersey.java.net/"
      jersey:generatedBy="Jersey: 2.0-SNAPSHOT ${buildNumber}"/>
   <grammars/>
   <resources base="http://localhost:9998/">
      <resource path="country/15">
         <method name="GET" id="getCountry">
            <response>
               <representation mediaType="application/xml"/>
            </response>
         </method>
         <method name="OPTIONS" id="apply">
            <request>
               <representation mediaType="*/*"/>
            </request>
            <response>
               <representation mediaType="application/vnd.sun.wadl+xml"/>
            </response>
         </method>
      </resource>
   </resources>
</application>
```

FIGURE 2.2 – Un exemple d'un fichier WADL

partage des données sur le web, tout en accédant à leurs sources de données via des services web, ce qui fournit une méthode bien documentée, et interopérable pour l'interaction avec les différentes sources données hétérogènes. Ces services sont appelés les services web d'accès aux données, ou services DaaS (Data-as-a-Service).

Ces services DaaS sont un type particulier des services cloud conçus pour la gestion des données dans les environnements cloud computing. Actuellement, Le cloud computing est la technologie la plus favorable par les entreprises et les organisations notamment celles qui ont un budget informatique très réduit, afin de bâtir une infrastructure physique puissante (traitement, stockage, etc), scalable, auto-reconfigurable et sans avoir besoin d'un investissement initiale. Une plateforme cloud computing peut être considérée comme étant un ensemble de services regroupés en quatre couches : IaaS (Infrastructure-as-a-Service), PaaS (Platform-as-a-Service), DaaS (Data-as-a-Service), et SaaS (Sfotware-as-a-Service).

Architecture des services DaaS

La figure 2.3 présente l'architecture des services DaaS [18]. Cette architecture est constituée de trois couches : la couche *Données*, la couche *Service*, et la couche *Mapping*.

La couche *Données* implique toute source de donnée qui peut être accédée par un service DaaS, telle que : les bases de données, les applications orientées données (e.g., CRM, ERP, etc.), d'autres services, etc. Ces sources de données sont décrites par différents modèles de représentation (e.g., Relationnel, XML, HTML, CSV, WSDL, WADL, etc.), et interrogées via différents langages et protocoles (e.g., SQL, XPath/XQuery, SOAP, etc.).

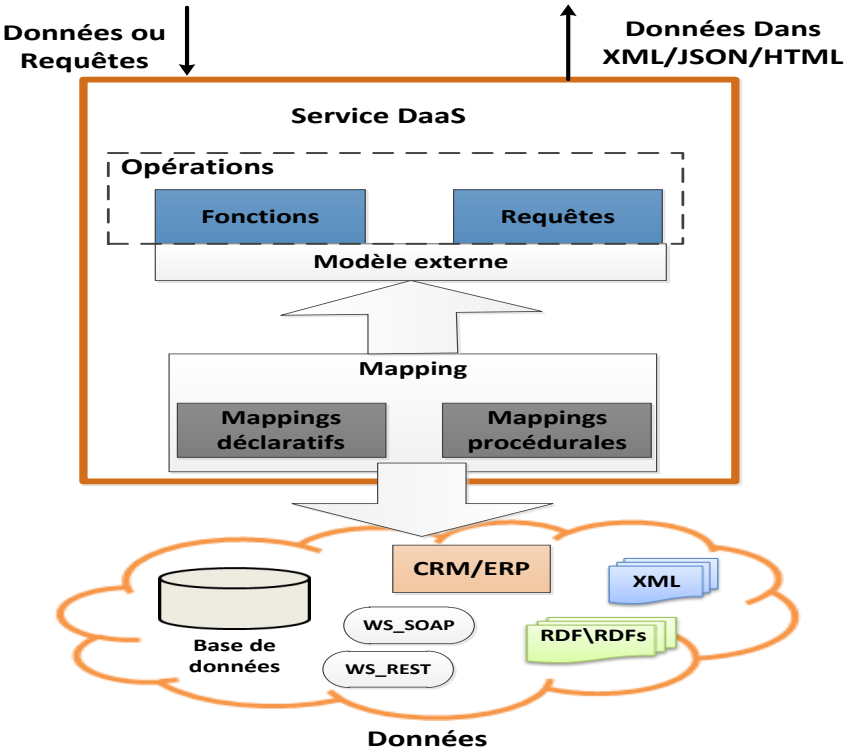


FIGURE 2.3 – Architecture des services DaaS

La couche *Service* constitue une interface uniforme qui permet aux utilisateurs d'accéder aux différentes sources de données, sans avoir besoins des connaissances liées à l'implémentation, et la localisation de celles-ci. Cette couche décrit l'ensemble des opérations à travers lesquelles un service DaaS est utilisé par ses clients. Nous définissons deux types d'opérations, les opérations de type *fonction*, et les opérations de type *requête*. Les opérations de type *fonction* sont similaires aux opérations d'un service web classique, dans lesquelles les utilisateurs sont complètement liés à la signature de l'opération, et à ce qui est retournée par cette dernière. En revanche, les opérations de type *requête* permettent aux utilisateurs de poser leurs propres requêtes, et d'interroger les sources de données de la façon qu'ils

désirent. Ces opérations prennent en paramètre une requête utilisateur décrite dans un langage de requête classique à savoir : SQL, XPath/XQuery, ou via un langage propriétaire, tel que : SOQL de Salesforce.com [7], OData de Microsoft [8], etc. Les données retournées par un service DaaS peuvent être décrites par plusieurs formats (e.g., HTML, JSON, XML, etc).

Les opérations d'un service DaaS et les requêtes utilisateurs (dans le cas des opérations de type *requête*) sont définies et formulées à travers le modèle de service (i.e., schéma de service) appelé : *modèle externe*. Le modèle externe est un modèle commun qui décrit de façon uniforme et globale les sources de données encapsulées par un service DaaS. Il peut être comparé au schéma global dans un système d'intégration de données classique (e.g., approche médiatrice, approche fédératrice, etc). Le modèle externe est défini lors de la construction de service en se basant sur les schémas des sources encapsulées. Il peut être créé automatiquement en utilisant un outil conçu pour la création des services DaaS (e.g., Oracle Data Service Integrator [9]). Le modèle externe est fourni aux utilisateurs de service en utilisant des fonctions spécifiques qui retournent sa description (sous forme d'un script SQL LDD, ou via un fichier XML).

Les correspondances entre les modèles des données encapsulées et le modèle externe du service DaaS sont décrites dans la couche *Mapping*. Le mapping associcié à une source de données définit comment le service l'accède. Il existe de types de mappings, les mappings déclaratifs, et les mappings procédurales. Les mappings déclaratifs sont souvent similaires aux vues d'une base de données relationnelle. Ils sont utilisés pour définir les mappings des sources de données dont le langage d'interrogation est déclaratif (e.g., SQL pour le relationnel, XQuery pour XML, SPARQL pour RDF/RDFs, etc). Les mappings procédurales sont utilisés pour les données qui sont fournies par des applications (e.g., CRM, ERP, etc), ou par des services DaaS.

Notons qu'un service DaaS peut effectuer différentes opérations (e.g., Interrogation, Suppression, Insertion, etc) sur les données qu'il les encapsule. Dans notre travail, nous nous intéressons aux services DaaS qui accèdent aux sources de données uniquement en mode lecture (Interrogation), ce que nous entendons par les services *sans-état* (*stateless*).

7. Salesforce.com, Inc. Salesforce.com Object Query Language (SOQL), 2010.
8. OData. Open Data Protocol, 2010. http ://www.odata.org/.
9. http ://www.oracle.com/technetwork/middleware/data-service-integrator/

2 Base de données probabiliste

Au cours des dernières années, la gestion des données incertaines a reçu une attention croissante surtout avec l'émergence de plusieurs nouvelles applications à savoir : les sensors (capteurs) [97], les réseaux RFID [57, 118], les systèmes de nettoyage et d'extraction des données, etc. L'incertitude est inhérente à ces applications, ce qui mène à une exigence extrême pour la gestion des données incertaines.

Plusieurs approches ont été proposées pour modéliser les données incertaines (i.e., le degré de certitude ou de confiance) à savoir, les approches à base de logique floue [12,17], les approches basées sur la théorie de Dempster-shafer [21], et les approches probabilistes [11, 24, 26, 36, 97], ou ce que nous entendons par les *bases de données probabilistes*. Cependant, la plupart des applications qui produisent des données incertaines quantifient ces dernières dans des bases de données probabilistes. Par exemple, les systèmes d'extraction de données sont basés sur des modèles probabilistes, i.e. ; les données qu'ils extraient sont probabilistes [48,66] ; Le nettoyage des flux RFID produisent également des distributions de probabilités [98] ; L'analyse de données dans la prévision financière reposent sur des modèles statistiques qui génèrent souvent des données probabilistes [56] ; etc.

2.1 Type d'incertitude

Deux types d'incertitudes sont utilisés dans les bases de données probabilistes : incertitude au niveau des tuples et incertitude au niveau des attributs.

Dans l'*incertitude au niveau des tuples*, nous ne savons pas si un tuple appartient à une base de données ou non. Dans une telle situation, chaque tuple t est considéré comme étant une variable aléatoire booléenne x (i.e. ; x a un domaine booléen), tel que, x=*true* lorsque t est présent, et $false$ s'il est absent. La figure 2.4(a) donne un exemple d'une table relationnel qui décrit la localisation des véhicules à un instant donné. L'incertitude dans cette table concerne les tuples. Cette incertitude provient de plusieurs sources, e.g. ; dans le tuple t_1, est ce que le véhicule **C1** est celui capté par le radar ou il s'agit d'un autre véhicule. Aussi, est ce que **L1** est la localisation exacte de **C1** à l'instant (**11 :23**), ou le GPS de ce véhicule a fait des erreurs, etc. Pour cette raison, nous associons à chaque tuple une probabilité qui définit son exactitude, e.g. ; t_1 avec 0.9, t_2 avec 0.2, etc.

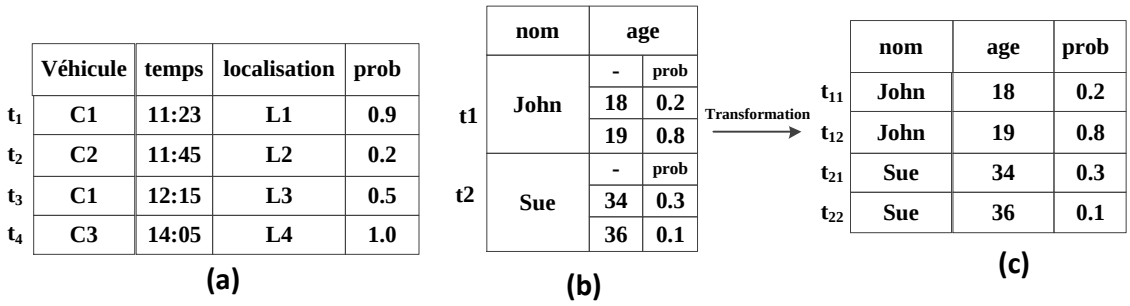


FIGURE 2.4 – **(a)** Incertitude au niveau des tuples. **(b)** Incertitude au niveau des attributs. **(c)** Transformation de l'incertitude au niveau des attributs vers celle des tuples.

Dans l'*incertitude au niveau des attributs*, la valeur d'un attribut A est incertaine. Autrement dit, pour chaque tuple t, l'attribut A est associé à une variable aléatoire dont le domaine est l'ensemble des valeurs possibles que l'attribut A peut prendre pour le tuple t. La figure 2.4(b) montre un exemple d'une incertitude au niveau des attributs. Dans cet exemple, l'âge d'une personne est incertain, e.g. ; est ce que **John** a 18 ou 19 ans.

Dans la pratique, l'*incertitude au niveau des attributs* peut être convertie en *incertitude au niveau des tuples*. Cette transformation est réalisée comme suit. Pour chaque tuple t dont l'attribut A prend les valeurs possibles a_1, a_2, a_3,..., nous créons plusieurs tuples t_1, t_2, t_3,..., qui sont identiques à t sauf pour l'attribut A, i.e. ; $t_1.A{=}a_1$, $t_2.A{=}a_2$, etc. De cette façon, chaque tuple t_i est incertain, et peut être décrit par une variable aléatoire booléenne. Cependant, les tuples obtenus sont mutuellement exclusifs. Une telle dépendance probabiliste doit être prise en considération par le modèle de représentation utilisé. Nous détaillons dans l'état de l'art (section 3.2) les différents modèles proposés pour représenter les bases de données probabilistes. La figure 2.4(c) montre comment l'incertitude au niveau des attributs dans la figure 2.4(b) est transformée en incertitude au niveau des tuples.

Dans le reste de ce chapitre, nous nous intéressons uniquement aux bases de données probabilistes manipulant des tuples incertains (i.e. ; *incertitude au niveau des tuples*).

2.2 Sémantique des bases de données probabilistes

Dans une base de données probabiliste, chaque tuple à une certaine probabilité qui décrit son degré d'appartenance ou d'existence. Les bases de données probabilistes utilisent

les principes de la théorie des probabilités en représentant les tuples incertains comme étant des événements probabilistes. Plus spécifiquement, chaque tuple t est un événement probabiliste, tel que t existe dans la base de données avec une probabilité qui est égale à $Pr(t)$, et il n'existe pas dans la base de données avec la probabilité complémentaire $Pr(\neg t) = 1 - Pr(t)$.

La probabilité d'un tuple t est toujours comprise dans l'intervalle $]0, 1]$ et qu'elle est interprétée comme étant son degré de confiance ou d'appartenance. En outre, un tuple avec une probabilité plus grande représente toujours un degré de confiance plus élevé. Par exemple, pour une requête donnée : un résultat avec une probabilité élevée est plus crédible qu'un résultat avec une faible probabilité.

Plusieurs approches [1, 25, 104] adoptent la sémantique des mondes possibles (*en anglais*, possible worlds semantics) pour interpréter les bases de données probabilistes. Une base de données probabiliste BDP est interprétée comme étant un ensemble de bases de données déterministes $pwd(BD)$=$\{BD_1, BD_2, ..., BD_n\}$ représentant tous les états (i.e. ; instances) possibles de BDP. L'ensemble des bases de données possibles correspond à toutes les combinaisons possibles des tuples dans BDP, tel que, chaque instance possible BD_k est un sous-ensemble de tuples, i.e. ; un tuple $t_i \in BDP$ est un tuple qui est impliqué dans au moins une base de données possible. S'il existe un tuple $t_i \in BDP$ dont $Pr(t) = 1$ alors dans ce cas, t_i est certain, et doit figurer dans tous les mondes possibles. Le nombre des mondes possible est exponentiel, tel que si le nombre de tuples incertains (leur probabilité ≤ 1) dans BDP est égale à m alors $|pwd(BD)|$=2^m.

Chaque monde possible $BD_k \in pwd(BD)$ (i.e. ; base de données possible) peut être vue comme étant une conjonction d'événements probabilistes dont la ***probabilité jointe*** est déterminée en fonction de la probabilité des événements existants (i.e. ; $Pr(t_i)$ si $t_i \in BD_k$), et la probabilité complémentaire des événements absents (i.e. ; $1 - Pr(t_i)$ si $t_i \notin BD_k$). De plus, le calcul de la probabilité d'un monde possible doit tenir compte des dépendances probabilistes (i.e. ; corrélations) qui existent entre les tuples. La somme des probabilités de tous les mondes possibles est égale toujours à 1, i.e. ; la base de données probabiliste définit une distribution de probabilité sur l'ensemble de ses instances possibles.

Définition 2.1 (***L'interprétation d'une base de données probabiliste***) :Une base

de données probabiliste est une distribution de probabilité BDP=($pwd(BD)$, Pr), où $pwd(BD)$=$\{BD_1, BD_2,..., BD_n\}$ est un ensemble fini de bases de données possibles, appelé mondes possibles, et Pr est une fonction de probabilité Pr : $pwd(BD) \longrightarrow]0, 1]$ tel que :$\sum_{BD_k \in pwd(BD)} Pr(BD_k)$=1.

Exemple La figure 2.5(b) montre les huits bases de données possibles qui correspondent à la base de données probabiliste dans la figure 2.5(a). Chaque base de données possible contient un sous-ensemble de tuples, e.g. ; BD_1 contient $\{t_1, t_2, t_3, t_4\}$, BD_3 contient $\{t_1, t_3, t_4\}$, etc. Chaque monde possible à une probabilité qui est calculée en fonction de la probabilité de tous les tuples présents(resp ; la probabilité complémentaire des tuples absents), ainsi que les dépendances qui existent entre eux. Par exemple, supposons que les tuples dans la base de données probabiliste sont indépendants (i.e. ; pas de dépendance probabiliste), alors dans ce cas, la probabilité de BD_1=$Pr(t_1)$*$Pr(t_2)$*$Pr(t_3)$*$Pr(t_4)$=0.09 ; $Pr(BD_3)$=$Pr(t_1)$*$[1 - Pr(t_2)]$* $Pr(t_3)$* $Pr(t_4)$=0.36 ($[1 - Pr(t_2)]$ puisque t_2 est absent dans BD_3) ; etc. Nous remarquons que le tuple t_4 figure dans tous les mondes possibles, ceci s'explique par le fait qu'il est certain (i.e. ; $Pr(t_4) = 1$), et par conséquent, les mondes possibles qui ne contiennent pas t_4 sont éliminés parce que leur probabilité est égale à 0. Par exemple, soit $\{t_1, t_2, t_3\}$ un monde possible qui ne contient pas t_4, la probabilité de celui-ci est égale à $Pr(t_1)$* $Pr(t_2)$* $Pr(t_3)$* $[1 - Pr(t_4)]$= 0.9* 0.2* 0.5* [1-1]=0, donc il sera éliminé.

	Véhicule	temps	localisation	prob
t_1	C1	11:23	L1	0.9
t_2	C2	11:45	L2	0.2
t_3	C1	12:15	L3	0.5
t_4	C3	14:05	L4	1.0

(a)

mondes possibles	prob
BD_1={t1,t2,t3,t4}	0.09
BD_2={t1,t2,t4}	0.09
BD_3={t1,t3,t4}	0.36
BD_4={t2,t3,t4}	0.01
BD_5={t1,t4}	0.36
BD_6={t2,t4}	0.01
BD_7={t3,t4}	0.04
BD_8={t4}	0.04
	$\sum$ 1

(b)

FIGURE 2.5 – **(a)** Base de données probabiliste **(b)** Ensemble des mondes possibles

Nous détaillons dans la partie état de l'art (chapitre 6) comment $Pr(BD_K)$ est calculée (i.e. ; probabilité d'un monde possible), parce que celle-ci dépend non seulement de la probabilité des tuples, mais aussi du modèle de représentation utilisé (i.e. ; les types de corrélations utilisés).

2.3 Sémantique des requêtes probabilistes

Dans le contexte des données incertaines, la sémantique des requêtes est la façon dont une requête est évaluée sur une base de données probabiliste. Rappelons que le résultat de l'évaluation d'une requête Q sur une base de données déterministe BD (i.e. ; certaine) est un ensemble de tuples, noté par $Q(BD)$.

Soit BDP une base de données probabiliste, et soit $pwd(BD)$=$\{BD_1, BD_2,..., BD_n\}$ l'ensemble des bases de données possibles de BDP. Evaluer une requête Q sur la base de données probabiliste BDP, revient à l'appliquer sur chaque base de données possible $BD_k \in pwd(BD)$, comme dans le cas certain (i.e. ; puisque chaque instance possible BD_k est déterministe). Cependant, le résultat obtenu par cette évaluation est interprété de deux façons différentes (i.e. ; deux sémantiques différentes) [114] : *Résultats possibles*, et *Tuples possibles*.

Sémantique des Résultats possibles

Dans la sémantique des *Résultats possibles*, la requête est appliquée sur chaque base de données possible $BD_k \in pwd(BD)$. Ensuite, le résultat obtenu est interprété comme étant un ensemble de résultats possibles, tel que, chacun contient 0 ou plusieurs tuples (i.e ; le résultat $\emptyset$ est inclus). Plus spécifiquement, un résultat possible correspond à l'ensemble des tuples retournés par l'évaluation de Q sur un ou plusieurs mondes possibles (i.e. ; lorsque $Q(BD_k)$= $Q(BD_h)$=...= $Q(BD_z)$). Par conséquent, l'évaluation de Q sur une base de données probabiliste $BDP = (pwd(BD), Pr)$ retourne une autre base de données probabiliste $Q(BDP)$=($pwd(R)$, Pr), tel que : l'ensemble des résultats possibles est considéré comme étant un ensemble de bases de données possibles $pwd(R)$=$\{R_1, R_2,..., R_m\}$=$\{Q(BD_1), Q(BD_2),..., Q(BD_n)\}$, et pour chaque résultat possible $R_h \in pwd(R)$ sa ***probabilité jointe*** est égale à la somme des probabilités des bases de données possibles dont le résultat correspond à R_h. Aussi, $Q(BDP)$ est appelée l'image de l'espace proba-

biliste BDP [47].

Il est important de noter que si le nombre des mondes possibles dans $pwd(BD)$=n, alors le nombre des résultats possibles est toujours inférieur ou égale à n, i.e. ; $|pwd(R)| \leq |pwd(BD)|$

Définition 2.2 (***Sémantique des Résultats possibles***) :Soit Q une requête, et soit BDP=($pwd(BD)$, P) une base de données probabiliste. La sémantique des **Résultats possibles** est une distribution de probabiliste $Q(BDP)$=($pwd(R)$, Pr), où :

- $pwd(R)$=$\{Q(BD_k) \mid BD_k \in pwd(BD)\}$,
- pour chaque $R_h \in pwd(R)$, $Pr(R_h)$=$\sum_{BD_k \in pwd(BD):Q(BD_k)=R_h} Pr(BD_k)$,
- $\sum_{R_h \in pwd(R)} Pr(R_h)$=1.

Exemple. Soit Q_1 (voir la figure 2.6(a)) une requête conjonctive qui a comme but de récupérer les véhicules dont la localisation a été saisie entre 11 :00 et 12 :00. Evaluer Q_1 à travers la base de données probabiliste dans la figure 2.6(b) revient à l'évaluer sur chaque mondes possibles dans la figure 2.6(c). Selon la sémantique des *Résultats possibles*, la requête accepte l'ensemble des résultats possibles dans la figure 2.6(d). Par exemple, évaluer Q_1 sur BD_1 et BD_2 retourne le même résultat R_1=$Q_1(BD_1)$=$Q_1(BD_2)$=$\{t_1, t_2\}$; R_2=$Q_1(BD_3)$=$Q_1(BD_5)$=$\{t_1\}$; etc. De plus, chaque résultat possible est associé à une probabilité qui égale à la somme des probabilités des mondes possibles qui le retourne. Dans notre exemple, R_1=$Q_1(BD_1)$=$Q_1(BD_2)$, par conséquent, la probabilité de R_1 est égale à $Pr(BD_1)$+ $Pr(BD_2)$=0.18.

Cette sémantique est particulièrement utile pour définir des vues sur les bases de données probabilistes. Dans ce cas, une requête est notée par V, et elle appelée une vue au lieu d'une requête. De plus, cette vue est considérée comme étant une fonction qui transforme une base de données probabiliste BDP vers une autre base de données probabiliste $V(BDP)$.

Sémantique des Tuples possibles

L'ensemble des *Résultats possibles* peut être très large, et il est impraticable de le retourner à l'utilisateur. Au lieu de cela, il est plus convenable de considérer qu'un seul

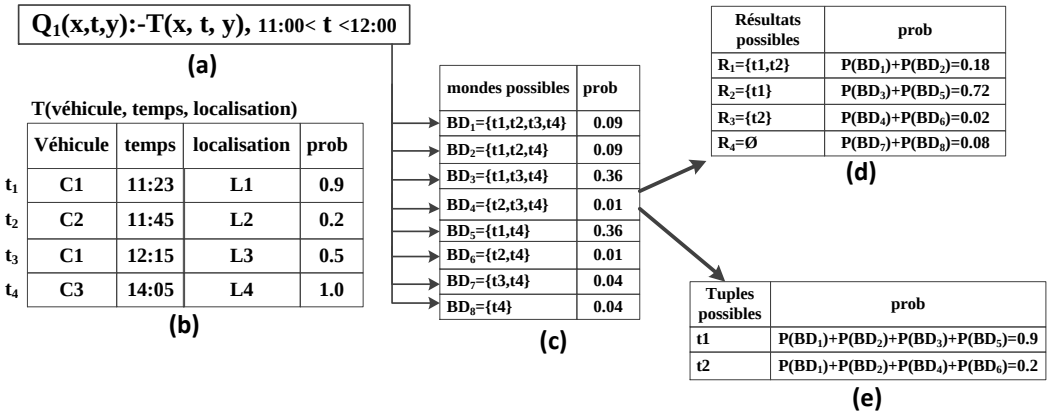


FIGURE 2.6 – **(a)** Une requête Q_1 **(b)** Base de données probabiliste **(c)** Ensemble des mondes possibles **(d)** Ensemble des *Résultats possibles* **(e)** Ensemble des *Tuples possibles*

résultat à la fois, ce que nous entendons par la sémantique des *Tuples possibles*. Dans cette sémantique, la requête est aussi appliquée sur chaque base de données possible, mais les résultats possibles sont combinés (i.e. ; en faisant l'union de tous les résultats), et un seul ensemble de tuples est retourné à l'utilisateur.

Définition 2.3 (***Sémantique des tuples possibles***) :Soit Q une requête, et soit $BDP = (pwd(BD), Pr)$ une base de données probabiliste. La sémantique des **Tuples possibles** est un ensemble de tuples probabilistes $Q(BDP)$=$\{(t_1, p_1), (t_2, p_2),...\}$, où :

- $t_1, t_2,... = \bigcup_{k=1}^{n} Q(BD_k)$, $BD_k \in pwd(BD)$
- Pour chaque tuple t_i sa **probabilité marginale** p_i=$\sum_{BD_k \in pwd(BD):t_i \in Q(BD_k)} Pr(BD_k)$.

Exemple. Nous continuons avec l'exemple précédent, mais en utilisant cette fois-ci la sémantique des *Tuples possibles*. La figures 2.6(e) montre que la requête Q_1 accepte 2 tuples possibles t_1 et t_2. Ces tuples sont obtenus en faisant l'union de tous les résultats possible, i.e. ; $Q_1(BD_1) \cup Q_1(BD_2) \cup,...,\cup Q_1(BD_8)$. Chaque tuple possible a une probabilité qui est égale à la somme des probabilités des mondes possibles dont le résultat contient ce tuple. Par exemple, t_1 appartient aux résultats obtenus par les mondes possibles BD_1 BD_2 BD_3 et BD_5 (i.e. ; $t_1 \in Q(BD_1)$, $t_1 \in Q(BD_2)$,...), par conséquent la probabilité marginale de t_1 est égale à $Pr(BD_1)+ Pr(BD_2)+Pr(BD_3)+ Pr(BD_5)$=0.9.

3 Réseaux bayésiens

Les *réseaux bayésiens* sont des modèles graphiques probabilistes conçus pour la représentation des variables aléatoires corrélées [23]. En outre, ils sont capables de représenter n'importe quelles dépendances probabilistes (corrélation complexe) qui peuvent exister entre les variables aléatoires. L'idée de base derrière les réseaux bayésiens est que les corrélations qui existent entre les variables aléatoires sont décrites par des distributions de probabilités conditionnelles. Un réseau bayésien est représenté par un graphe orienté acyclique, dans lequel les nœuds représentent les variables aléatoires, et les arcs représentent les corrélations orientées qui existent entre elles.

Définition 2.4 (***Réseau bayésien***) : Un réseau bayésien BN est un triplé $(X,E,\mathcal{P})$ où :

- $X=\{x_1,...,x_n\}$ est un ensemble de variables aléatoires, tel que, chaque variable x_i a un domaine $dom(x_i)$ qui définit ses valeurs possibles.
- E est un ensemble d'arcs. Si deux variables aléatoires x_i et x_j sont corrélées, alors il existe un arc $(x_i,\ x_j)\in E$.
- $\mathcal{P}$ est un ensemble de distributions de probabilités conditionnelles, $\mathcal{P}=\{Pr(x_i|\ par(x_i)) : x_i \in X\}$, où $par(x_i)$ est l'ensemble des parents de x_i (i.e., prédécesseurs directes) [10].

Définition 2.5 (***Distributions de probabilités conditionnelles***) : Soit BN $(X,E,\mathcal{P})$ un réseau bayésien, et soit $x_i\in X$ une variable aléatoire, tel que $par(x_i)=\{x_k,...,x_h\}$ est l'ensemble des prédécesseurs de x_i dans le graphe de BN. Soit $Pr(x_i|\ par(x_i))\in \mathcal{P}$ une distribution de probabilité conditionnelle qui dénote comment la valeur de x_i dépend de celles de ses parents :

$$\sum_{ax_i\in dom(x_i)} Pr(x_i{\sim}ax_i|par(x_i){\sim}AX_{par})=1,\ \forall\ AX_{par}\in dom(x_k)\times ...\times dom(x_h).$$

où AX_{par} est un assignement possible de toutes les variables dans $par(x_i)$.

Example. Soit $\{x_1,x_2,x_3\}$ un ensemble de variables aléatoires booléennes (i.e., pour chaque x_i, $dom(x_i)=\{0,1\}$). Chaque variable a une probabilité marginale : $Pr_{x_1}=0.3$, $Pr_{x_2}=0.7$, et $Pr_{x_3}=0.8$. Supposons qu'il existe des corrélations entre ces variables, tel que : $x_1 \oplus x_2$ (i.e., x_1 et x_2 sont mutuellement exclusives), et $x_1 \rightarrow x_3$ (i.e., la présence

10. Variables sans parents (i.e., nœuds racines) sont associées à leurs probabilités marginales.

de x_1 implique la présence de x_3). Ces variables corrélées peuvent être représentées par un réseau bayésien $BN_1(X, E, \mathcal{P})$ (voir la figure 2.7), où : X=$\{x_1, x_2, x_3\}$, E=$\{(x_1, x_2), (x_1, x_3)\}$, $\mathcal{P}$=$\{Pr(x_1), Pr(x_2|x_1), Pr(x_3|x_1)\}$. Le nœud de la variable x_1 n'a pas de prédécesseur dans le graphe de BN_1, par conséquent il est associé avec la probabilité marginale $Pr_{x_1} = 0.3$. Les autres nœuds x_2 et x_3 maintiennent des tables de probabilités conditionnelles qui représentent leurs distributions conditionnelles sachant leur parent x_1. Les tables de probabilités conditionnelles sont déduites à partir de la sémantique des corrélations. Par exemple, pour la corrélation $(x_1 \oplus x_2)$, nous avons :

- $Pr(x_1{=}1 \wedge x_2{=}1){=}0$,
- $Pr(x_1{=}1 \wedge x_2{=}0){=}Pr_{x_1}$,
- $Pr(x_1{=}0 \wedge x_2{=}1){=}Pr_{x_2}$,
- $Pr(x_1{=}0 \wedge x_2{=}0)$=1-$[Pr_{x_1} + Pr_{x_2}]$.

Par conséquent, la distribution conditionnelle $Pr(x_2|x_1)$ est calculée comme suit :

- $Pr(x_2{=}1|x_1{=}1){=}\frac{Pr(x_1=1 \wedge x_2=1)}{Pr(x_1=1)}{=}\frac{0}{Pr_{x_1}}{=}0$;
- $Pr(x_2{=}0|x_1{=}1){=}\frac{Pr(x_1=1 \wedge x_2=0)}{Pr(x_1=1)}{=}\frac{Pr_{x_1}}{Pr_{x_1}}{=}1$;
- $Pr(x_2{=}1|x_1{=}0){=}\frac{Pr(x_1=0 \wedge x_2=1)}{Pr(x_1=0)}{=}\frac{Pr_{x_2}}{1-Pr_{x_1}}{=}1$;
- $Pr(x_2{=}0|x_1{=}0){=}\frac{Pr(x_1=0 \wedge x_2=0)}{Pr(x_1=0)}{=}\frac{1-[Pr_{x_1}+Pr_{x_2}]}{1-Pr_{x_1}}{=}\frac{0}{1-0.3} = 0$.

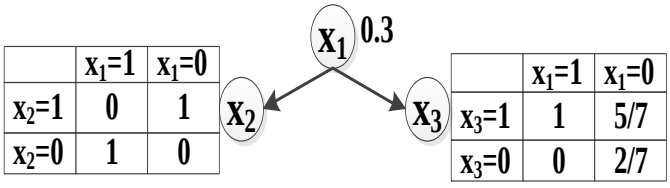


FIGURE 2.7 – Exemple d'un réseau bayésien

Pour la corrélation $(x_1 \rightarrow x_3)$, nous avons :

- $Pr(x_1{=}1 \wedge x_3{=}1){=}Pr_{x_1}$,
- $Pr(x_1{=}1 \wedge x_3{=}0){=}0$,
- $Pr(x_1{=}0 \wedge x_3{=}1){=}Pr_{x_3} - Pr_{x_1}$,
- $Pr(x_1{=}0 \wedge x_3{=}0){=}1 - Pr_{x_3}$.

Par conséquent, la distribution conditionnelle $Pr(x_3|x_1)$ est calculée comme suit :

- $Pr(x_3{=}1|x_1{=}1){=}\frac{Pr(x_1=1 \wedge x_3=1)}{Pr(x_1=1)}{=}\frac{Pr_{x_1}}{Pr_{x_1}}{=}1$;
- $Pr(x_3{=}0|x_1{=}1){=}\frac{Pr(x_1=1 \wedge x_3=0)}{Pr(x_1=1)}{=}\frac{0}{Pr_{x_1}}{=}0$;

- $Pr(x_3{=}1|x_1{=}0){=}\frac{Pr(x_1=0\wedge x_3=1)}{Pr(x_1=0)}{=}\frac{Pr_{x_3}-Pr_{x_1}}{1-Pr_{x_1}}{=}\frac{0.5}{1-0.3} = \frac{5}{7}$;
- $Pr(x_3{=}0|x_1{=}0){=}\frac{Pr(x_1=0\wedge x_3=0)}{Pr(x_1=0)}{=} \frac{1-Pr_{x_3}}{1-Pr_{x_1}}{=}\frac{0.2}{1-0.3} = \frac{2}{7}$.

L'ensemble des variables aléatoires X dans un réseau bayésien BN peut être associé à plusieurs assignements complets qui assignent chaque variable par une de ses valeurs possibles. La probabilité jointe d'un assignement complet AX_k est calculée en multipliant les valeurs retournées par toutes les distributions conditionnelles (sachant AX_k) dans $\mathcal{P}$. La somme des probabilités jointes de tous les assignements complets de l'ensemble X est égale à 1.

Définition 2.6 (***Probabilité jointe***) : Soit $BN(X,\ E,\ \mathcal{P})$ un réseau bayésien, et soit $AX{=}\{AX_1,...,AX_n\}$ l'ensemble de tous les assignements complets de X, tel que chaque AX_k assigne chaque variable $x_i{\in}X$ avec une valeur possible $ax_i \in dom(x_i)$. La probabilité jointe est définie comme suit :

- $Pr_{BN}(X \sim AX_k){=}\prod_{x_i\in X} Pr(x_i|\ par(x_i))$. (chaque x_i est assignée par $ax_i \in AX_k$).
- $\sum_{AX_k\in AX} Pr_{BN}(X \sim AX_k) = 1$.

Example. Nous continuons avec notre exemple. L'ensemble des variables aléatoires$\{x_1,\ x_2,\ x_3\}$ dans le réseau bayésien BN_1 peut avoir plusieurs assignements complets, e.g., AX_1 :$\{x_1{=}1,\ x_2{=}0,\ x_3{=}1\}$, AX_2 :$\{x_1{=}1,\ x_2{=}0,\ x_3{=}0\}$, AX_3 :$\{x_1{=}0,\ x_2{=}1,\ x_3{=}1\}$, etc. La probabilité jointe de chaque assignement complet est calculée en multipliant les valeurs retournées par les tables de probabilités conditionnelles lorsque en appliquant celui-ci. Nous montrons ci-dessous comment la probabilité jointe d'un assignement complet est calculée :

$$Pr_{BN1}(X \sim AX_1) = Pr(x_1 = 1) \times Pr(x_2 = 0|x_1 = 1) \times Pr(x_3 = 1|x_1 = 1)$$
$$= \quad 0{,}3 \quad \times \quad 1 \quad \times \quad 1 \quad = 0{,}3.$$

$$Pr_{BN1}(X \sim AX_2) = Pr(x_1 = 1) \times Pr(x_2 = 0|x_1 = 1) \times Pr(x_3 = 0|x_1 = 1)$$
$$= \quad 0{,}3 \quad \times \quad 1 \quad \times \quad 0 \quad = 0.$$

$$Pr_{BN1}(X \sim AX_3) = Pr(x_1 = 0) \times Pr(x_2 = 1|x_1 = 0) \times Pr(x_3 = 1|x_1 = 0)$$
$$= \quad 0{,}7 \quad \times \quad 1 \quad \times \quad \tfrac{5}{7} \quad = 0{,}5.$$

4 Conclusion

Dans ce chapitre, nous avons présenté les principaux concepts autour de la technologie des services Web, ainsi que les services web d'accès aux données. Nous avons également présenté les modèles probabilistes que nous devrons utiliser pour résoudre le problème d'incertitude lié à la sémantique des services DaaS. Maintenant, le lecteur devrait être en mesure de comprendre nos contributions décrites dans les chapitres suivants ainsi que le reste de cette thèse.

Chapitre 3

Modélisation et composition des services à sémantique incertaine

1 Introduction

Dans les systèmes d'intégration de données à base de services web sémantiques, les sources de données locales sont décrites par des services DaaS permettant un traitement plus dynamique et plus flexible. En outre, l'interface à travers laquelle ces sources de données sont interrogées est basée sur une ontologie médiatrice, et les requêtes utilisateurs sont définies de façon déclarative (e.g., requête SPARQL). La relation sémantique entre un service DaaS et l'ontologie médiatrice est définie comme étant une vue sémantique décrivant l'interprétation des données encapsulées par ce service. Les vues sémantiques sont utilisées pour reformuler (réécrire) les requêtes utilisateurs en des compositions de services.

Dans la plupart des cas, un service DaaS est associé à une et une seule vue sémantique. Cela se produit lorsque les annotateurs, i.e., les personnes chargées de la définition des vues sémantiques, sont en mesure d'interpréter correctement les données encapsulées par un service DaaS, et peuvent par conséquent définir avec certitude la vue sémantique correspondante à celui-ci. Cependant, dans certains cas, la sémantique des données encapsulées par un service est imprécise et peut donc accepter différentes interprétations possibles. Le service doit ensuite être associé à plusieurs vues sémantiques (probables)

représentant ses différentes interprétations possibles. Cela peut se produire pour différentes raisons. Par exemple, l'annotateur et le fournisseur de service pourraient être deux entités indépendantes, où l'annotateur n'a pas suffisamment d'informations sur le service publié (e.g., Meta-data, description textuelle, etc), et cela afin de définir la sémantique du service avec certitude, ou lorsque le fournisseur lui-même n'a pas des connaissances approfondies sur les données accédées par son service. Ce dernier cas est particulièrement courant dans le contexte des données ouvertes où souvent la sémantique des données publiées est imprécise voire inexacte. Dans tous ces cas, les techniques de *Schema Matching* [69, 106] pourraient être exploitées pour obtenir les sémantiques possibles d'un service (i.e., l'ensemble des vues sémantiques possibles), ainsi que leurs degrés de certitude (i.e., probabilités).

1.1 Exemple de motivation

Considérons les services DaaS de la Figure 3.1, qui sont des services typiques pouvant être construits en utilisant des sources de données ouvertes (e.g., *university membership dataset*[1], *US CMS active research projects datasets*[2], etc) et fournis par des systèmes tels que *opendata.socrata.com, programmableweb,..* etc. Les symboles "$" et " ?" dénotent respectivement, les entrés et les sorties d'un service DaaS. Dans cette partie nous supposons que les services sont indépendants, i.e., ils n'encapsulent pas les mêmes sources de données.

Les annotateurs ou les fournisseurs de services peuvent définir les vues sémantiques de ces services (à travers une ontologie de domaine, voir la Figure 3.2) en utilisant un système de matching semi-automatique. Cependant, dans cet exemple, ils n'arrivent pas à interpréter correctement la sémantique de ces services (i.e., sémantique incertaine), ce qui les empêche de définir pour chacun une seule vue sémantique (voir la Figure 3.3). Par exemple, ils ne peuvent pas savoir si le service s_1 renvoie des anciens chercheurs ou les chercheurs actuels d'une université donnée, parce que ceci n'est pas précisé dans la source de données encapsulée par s_1 (i.e., *university membership dataset*). Par conséquent, s_1 est associé à deux vues sémantiques v_{11} et v_{12} (voir Figure 3.3). La sémantique du service s_2 est également incertaine, tel que s_2 est associé à trois vues sémantiques v_{21}, v_{22} et v_{23}. La première vue v_{21} interprète les sorties $?c_1$ et $?c_2$ respectivement comme étant des

1. http ://www.data.gouv.fr/fr/dataset/les-membres-de-l-institut-universitaire-de-france.
2. https ://catalog.data.gov/dataset/active-projects-report-fe965

Services DaaS	Fonctionnalités
s_1(*$u, ?r*)	Retourne les noms des chercheurs (*?r*) pour une université donnée (*$u*).
s_2(*$r, ?c1, ?c2*)	Retourne les contactes (*?c1, ?c2*) d'un chercheur (*$r*).
s_3(*$r, ?a, ?t, ?d*)	Retourne les différentes informations (?a, ?t,?d) (âge, contacte, adresse) d'un chercheur (*$r*).
s_4(*$t, ?o*)	Retourne l'opérateur téléphonique (*?o*) d'un numéro de téléphone (*$t*).

FIGURE 3.1 – Ensemble de services DaaS

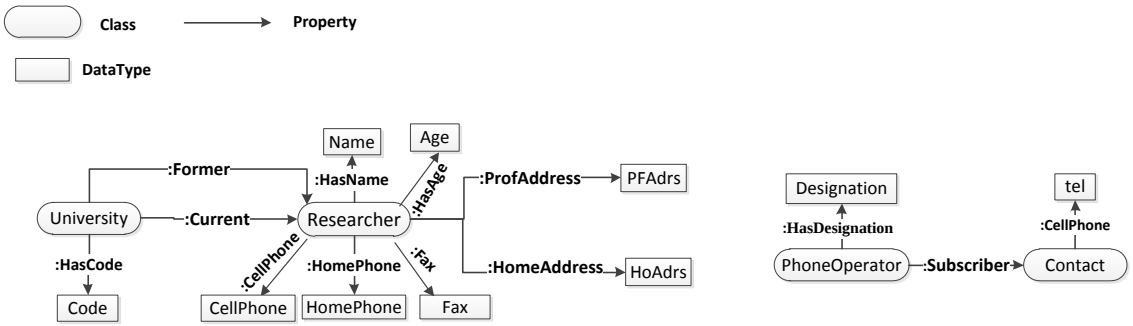


FIGURE 3.2 – Exemple d'une ontologie de domaine.

numéros de téléphone mobile et des numéros de fax, ; v_{22} interprète ces mêmes sorties comme étant des numéros de téléphone domicile et des numéros de téléphone mobile, v_{23} les interprète comme étant des numéros de téléphone domicile et des numéros de fax. s_4 a une interprétation certaine (i.e., déterministe) et par conséquent il est associé à une seule vue sémantique v_{41}.

1.2 Challenges

Supposons qu'un utilisateur *Alice* veut trouver les numéros de téléphone mobile et les opérateurs téléphoniques des chercheurs de l'université d'Irvine. La requête d'Alice, i.e., Q_1 *: "Retourner les numéros de téléphone mobile ainsi que de leurs opérateurs téléphonique des chercheurs actuels de l'université d'Irvine*" peut être résolue en utilisant les services de notre exemple de motivation. Cependant, composer des services à sémantique incertaine pour répondre à une requête comme Q_1 soulève les challenges suivants :

– *Comment modéliser la sémantique incertaine des services DaaS* : Un nouveau modèle de représentation pour les services DaaS est nécessaire. Ce nouveau modèle doit être capable de représenter un service avec ses vues sémantiques possibles ainsi que leurs

Vues sémantiques possibles de s_1($u, ?r)		Description
v_{11}	Select ?r Where{?U :rdftype :University ?U :hasCode $u, ?U ***:current*** ?T ?T :rdftype : Researcher, ?T :HasName ?r	Retourne les noms (?r) des chercheurs *actuels* à l'université ($u)
v_{12}	Select ?r Where{?U :rdftype :University ?U :HasCode $u, ?U ***:former*** ?T ?T :rdftype :Researcher, ?T :HasName ?r	Retourne les noms (?r) des *anciens* chercheurs de l'université ($u)

Vues sémantiques possibles de s_2($r, ?c1, ?c2)		Description
v_{21}	Select ?c1, ?c2 Where{?T :rdftype :Researcher, ?T :hasName $r ?T :*cellPhone* ? c1 ?T :*fax* ?c2	Retourne le *téléphone mobile* (?c1), et le *fax* (?c2), d'un chercheur ($r)
v_{22}	Select ?c1, ?c2 Where{?T :rdftype :Researcher, ?T :hasName $r ?T :*homePhone* ?c1 ?T :*cellPhone* ?c2	Retourne le *téléphone domicile* (?c1), et le *téléphone mobile* (?c2), d'un chercheur ($r)
v_{23}	Select ?c1, ?c2 Where{?T :rdftype :Researcher, ?T :hasName $r, ?T :*homePhone* ?c1 ?T :*fax* ?c2	Retourne le *téléphone domicile* (?c1), et le *fax* (?c2), d'un chercheur ($r)

Vues sémantiques possibles de s_3($r, ?a, ?t, ?d)		Description
v_{31}	Select ?a, ?t, ?d Where{ ?T :rdftype :Researcher, ?T :hasName $r, ?T :hasAge ?a, ?T :*homePhone* ?t, ?T :*homeAddress* ?d	Retourne l'âge (?a), le *téléphone domicile* (?t), et l'*adresse personnelle* (?d), d'un chercheur ($r)
v_{32}	Select ?a, ?t, ?d Where{?T :rdftype :Researcher, ?T :hasName $r; ?T :hasAge ?a, ?T :*CellPhone* ?t ?T :*ProfAddress* ?d	Retourne l'âge (?a), le *téléphone mobile* (?t), et l'*adresse professionnelle* (?d), d'un chercheur ($r)

Vues sémantiques possibles de s_4($t, ?o)		Description
v_{41}	Select ?o Where {?P :rdftype :PhoneOperator, ?P :HasDesignation ?o, ?P :Subscriber ?A ?A :rdftype :Contact; ?A :CellPhone $t	Retourne l'opérateur téléphonique (?o) d'un numéro de numéro de téléphone ($t)

FIGURE 3.3 – Les vues sémantiques possibles de chaque service.

degrés de certitude ou d'exactitude (i.e., probabilité). De plus, il doit prendre en considération les dépendances (i.e., corrélations) qui peuvent exister entre les vues sémantiques.

– *Comment interpréter une composition impliquant des services dont la sémantique est incertaine* : Les services sont souvent composés pour créer des services composites répondant à des besoins complexes. Quand une composition implique des services dont la sémantique est incertaine, alors dans ce cas, quelles sont les interprétations possibles de cette composition et quelles sont leurs probabilités ?.

– *Comment évaluer une requête utilisateur à travers des services incertains* : Contrairement au techniques de réécriture de requêtes classiques où les services et les compositions sont certaines, quand la sémantique des services devient incertaine, les techniques de réécriture de requêtes doivent (être étendues) retourner également les probabilités des réécritures obtenues (i.e., compositions).

1.3 Contributions

Dans ce chapitre, nous proposons une approche probabiliste [74] pour la modélisation et la représentation des services à sémantique incertaine, ainsi qu'une méthode de réécriture de requêtes. Dans notre approche, un service DaaS avec une sémantique incertaine est

sémantiquement décrit par un ensemble de vues sémantiques possibles, chacune avec une probabilité représentant son degré de certitude. Les services ainsi que leurs vues sémantiques possibles sont représentés dans un registre de services probabiliste (noté PSR) qui respect le modèle probabiliste Bloc-Indépendant-Disjoint (noté BID). Le registre de services probabiliste est interprété en se basant sur la sémantique des mondes possible (en anglais, *possible worlds semantics*). Nous définissons une nouvelle sémantique pour l'évaluation des requêtes à travers les registres de service probabilistes, et nous proposons un algorithme qui permet d'assurer une évaluation efficace des requêtes utilisateurs. Nous résumons ci-dessous nos principales contributions :

- *Registre de services probabiliste* : nous proposons un modèle probabiliste pour les services à sémantique incertaine. Dans notre modèle, chaque interprétation possible d'un service est représentée par une vue sémantique avec une probabilité. En outre, les services et leur interprétations possibles sont représentés dans un registre de services probabiliste (noté PSR) qui suit le modèle BID, de telle sorte que chaque tuple $<s_i, v_{ij}, Pr_{ij}>$ dans PSR signifie que le service s_i peut être interprété selon la vue sémantique v_{ij} avec une probabilité qui est égale à Pr_{ij}. Le registre de services probabiliste est partitionné en un ensemble de blocs, où les vues sémantiques du même bloc sont associées au même service, alors que les vues sémantiques qui appartiennent au même bloc sont associées avec des services DaaS différents. La sémantique (i.e., l'interprétation) de PSR est une distribution de probabilité sur un ensemble de registres de services possibles (i.e., registres de services déterministes).
- *Interprétation probabiliste de composition de services* : en se basant sur le concept registre de services probabiliste, l'interprétation d'une composition existante est définie comme étant un ensemble d'interprétations possibles obtenues en évaluant la même composition sur tous les registres de services possibles de PSR. Il est clair que l'évaluation de l'interprétation d'une composition est intractable. Ceci s'explique par le fait que le nombre de registres de services possibles est exponentiel. Dans ce sens, nous proposons un théorème qui permet de calculer efficacement (i.e., dans un temps polynomial) la probabilité d'une interprétation possible.
- *La sémantique d'évaluation des requêtes* : Evaluer une requête utilisateur Q à travers un registre de services probabiliste revient à l'évaluer sur chaque registre de services

possible (i.e., registre de services déterministe). L'évaluation de la requête peut accepter deux interprétation : la *Sémantique des Résultats Possibles* où les compositions possibles dans chaque monde possible sont retournées (associées avec la probabilité du monde), et la *Sémantique des Compositions possibles*, où les compositions possibles dans tous les mondes sont retournées avec leurs probabilités (agrégées).

– *Evaluation efficace des requêtes à travers les registres de services probabilistes* : Comme le nombre de registres de services possibles peut être extrêmement large, par conséquent l'évaluation des requêtes devient difficile voire impossible. Dans ce sens, nous proposons un algorithme qui permet d'évaluer efficacement les requêtes utilisateurs (i.e., dans un temps raisonnable), et sans avoir besoin de matérialiser l'ensemble des registres de services possibles. De plus, le résultat obtenu (i.e., compositions et leurs probabilités) par notre solution est en accordance avec la *Sémantique des Compositions possibles*.

Le reste de ce chapitre est organisé comme suit. Dans la section 2, nous décrivons les différents types de la sémantique incertaine. Dans la section 3, nous présentons le concept registre de services probabiliste et nous définissons sa sémantique. Dans la section 4, nous décrivons l'interprétation d'une composition dont les services impliqués sont incertains. Dans la section 5, nous définissons la sémantique de l'évaluation d'une requête à travers un registre de services probabiliste. La section 6 présente une approche qui permet l'évaluation efficace des requêtes. Dans la section 7, nous proposons quelques optimisations qui permettent d'améliorer l'efficacité de notre approche. Enfin, dans la section 8, nous concluons ce chapitre en invoquant les perspectives.

2 Différents types de la sémantique incertaine

Comme expliqué dans l'exemple de motivation, il n'est pas toujours possible de définir la sémantique d'un service DaaS avec exactitude et certitude. Cela arrive souvent lorsque le fournisseur de services ne fournit pas suffisamment d'informations (i.e., informations textuelle, métadonnées, etc) sur les sources de données accédées par ses services, ce qui mène à différentes interprétations possibles (i.e., la sémantique) pour le même service. Les différentes interprétations possibles d'un service DaaS doivent être exploitées pour

améliorer la description du service (e.g., le fichier WSDL pour les service SOAP, fichier WADL pour les service REST, etc), afin que les consommateurs de ce dernier puissent l'interpréter et l'utiliser correctement.

Dans cette thèse, nous adoptons une approche déclarative [9] pour décrire la sémantique des services DaaS, i.e., la sémantique d'un service est modélisée comme étant une vue sémantique (*vue RDF*) à travers une ontologie de domaine. Les différentes interprétations possibles d'un service DaaS sont représentées par un ensemble de vues sémantiques. Nous notons l'ensemble des vues sémantiques possibles d'un service s_i par $psv(s_i) = \{v_{i1}, v_{i2}, ..., v_{in}\}$, i.e., s_i est associé à n vues sémantiques possibles. Cependant, associer à un service plusieurs vues sémantiques signifie deux choses : *Incertitude liée aux schémas* et *Incertitude liée aux données* , i.e., l'incertitude liée à la sémantique des services est interprétée de deux façons différentes.

2.1 Incertitude liée aux schémas

Dans l'*Incertitude liée aux schémas*, les données retournées par un service sont toutes sémantiquement homogènes. Ceci permet l'interprétation de l'incertitude au niveau du schéma plutôt qu'au niveau des données, tout en définissant une seule vue sémantique qui décrit l'interprétation de toutes les données encapsulées. Autrement dit, la sémantique d'un service dans le cas de l'*Incertitude liée aux schémas* est *constante* et *incertaine*. Dans ce cas, une seule vue sémantique dans l'ensemble $psv(s_i)$ est correcte et corresponde à toutes les données retournées. En d'autre termes, la sémantique d'un service peut correspondre à l'un des mondes possibles (i.e., ici un monde possible signifie une vue sémantique).

Définition 3.1 (***Incertitude liée aux schémas***) : Soit s_i un service DaaS qui encapsule une source de données D_i. Soit $psv(s_i)$ un ensemble de vues sémantiques qui décrivent les différentes interprétations du s_i. $psv(s_i)$ est de type *Incertitude liée aux schémas* pour s_i $\rightarrow \exists!$ $v_{ij} \in psv(s_i)$ tel que ($\forall$ tuple $t_k \in D_i$ s.t. t_k est sémantiquement décrit par v_{ij}).

2.2 Incertitude liée aux données

Dans l'*Incertitude liée aux données*, les données retournées par un service sont sémantiquement hétérogènes, et chaque sous-ensemble de données a une sémantique particulière. Par exemple,

si un service retourne les contacts des clients : certains clients vont utiliser leur email comme contact, alors que d'autres vont choisir leur numéro de téléphone mobile, i.e., les données encapsulées par ce service sont sémantiquement divisées en deux sous-ensembles hétérogènes. Dans ce cas, l'incertitude se produit au niveau des données (i.e., pas comme les données incertaines), et différentes vues sémantiques sont applicables (i.e., partiellement correctes), où chacune correspond à un sous-ensemble de données. En d'autres termes, la sémantique du service dans l' *Incertitude liée aux données* est inconstante et incertaine, et peut correspondre à tous les mondes possibles.

Définition 3.2 (***Incertitude liée aux données***) : Soit s_i un service DaaS qui encapsule une source de données D_i. Cet ensemble de données contient d sous-ensembles de tuples $\{\grave{D}_{i1},\ldots,\grave{D}_{in}\}$. Soit $psv(s_i)$ un ensemble de vues sémantiques qui décrivent les différentes interprétations du s_i. $psv(s_i)$ est de type *Incertitude liée aux données* pour $s_i \rightarrow \exists!$ une collection de vues sémantiques $Col_h \subseteq psv(s_i)$ tel que, pour chaque sous-ensemble $\grave{D}_{ij} \subseteq D_i$ ($\forall$ tuple $t_k \in \grave{D}_{ij}$ s.t. t_k est sémantiquement décrit par la $j^{éme}$ vue v_{ij} dans Col_h).

3 Représentation des services à sémantique incertaine

Dans cette section, nous présentons une approche probabiliste pour la représentation des services DaaS à sémantique incertaine. Plus précisément, nous exploitons les principes des bases de données probabilistes pour mettre en place un modèle robuste pour les services incertains (i.e., registre de service).

3.1 Modèle probabiliste pour la sémantique incertaine

Dans notre travail, nous nous intéressons uniquement à l'incertitude liée aux schémas. Le deuxiéme type d'incertitude (i.e., Incertitude liée aux données) n'est pas pris en considération. Nous supposons que les tuples retournés par un service certain (i.e., déterministe) sont tous sémantiquement homogènes (i.e., qu'ils adhèrent tous à une seule vue sémantique possible).

Dans notre approche, chaque vue sémantique possible v_{ij} d'un service s_i est considérée comme étant un événement probabiliste, et par conséquent, elle est associée avec une

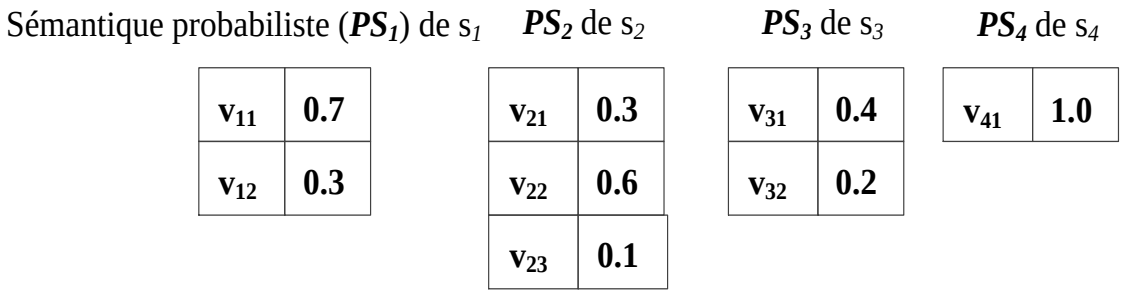


FIGURE 3.4 – Les sémantiques probabilistes des services de l'exemple de motivation.

probabilité représentant le degré de certitude pour que v_{ij} décrive la sémantique de s_i. Ce degré de certitude est quantifié par une valeur de probabilité $Pr(v_{ij})$, i.e., plus $Pr(v_{ij})$ est grand, plus v_{ij} est plausible. L'ensemble des vues sémantiques possibles avec leurs probabilités est appelé *Sémantique Probabiliste* d'un service.

Définition 3.3 (***Sémantique Probabiliste***) : Soit $S = \{s_1, s_2, ..., s_m\}$ un ensemble de services Daas, et soit MO une ontologie de domaine. Soit PS_i une *Sémantique Probabiliste* pour $s_i \in S$. Cette sémantique probabiliste est un espace probabiliste $PS_i = (psv(s_i), Pr)$, où $psv(s_i) = \{v_{i1}, v_{i2}, ..., v_{in}\}$ est un ensemble non vide de vues sémantiques, et Pr est une fonction de probabilité $Pr : psv(s_i) \rightarrow]0, 1]$, tel que pour chaque $v_{ij} \in psv(s_i), Pr(v_{ij})$ spécifie son degré de certitude.

Exemple. La Figure 3.4 représente les *Sémantiques Probabilistes* des services de l'exemple de motivation (les vues sémantiques sont celles de la Figure 3.3). A chaque vue sémantique possible v_{ij} est associée une probabilité. Par exemple, les probabilités des vues sémantiques v_{11} et v_{12} du service s_1 sont respectivement "0,7" et "0,3".

La sémantique probabiliste PS_i d'un service s_i peut être interprétée de deux façons différentes : *Sémantique Probabiliste non-exhaustive* et *Sémantique Probabiliste exhaustive.*

Dans le cas de la *Sémantique Probabiliste non-exhaustive*, la somme des probabilités des vues sémantiques associées à un service DaaS est strictement inférieure à 1. Ceci s'explique par le fait qu'un service peut accepter d'autres vues sémantiques (une ou plusieurs) qui ne sont pas décrites dans l'ensemble $psv(s_i)$.

Définition 3.4 (***Sémantique Probabiliste non-exhaustive***) : Soit $S = \{s_1, s_2, ..., s_m\}$ un ensemble de services Daas, et soit MO une ontologie de domaine. Soit $PS_i = (psv(s_i), Pr)$ une *Sémantique Probabiliste* pour $s_i \in S$ (formulée à travers MO). PS_i est une *Sémantique Probabiliste non-exhaustive* $\Longleftrightarrow \exists$ un ensemble non vide de vues sémantiques $\alpha \not\subseteq psv(s_i)$ tel que :

- $\sum_{v_{ij} \in psv(s_i)} Pr(v_{ij}) < 1$.
- $\sum_{v_{ik} \in \alpha} Pr(v_{ik}) = 1 - (\sum_{v_{ij} \in psv(s_i)} Pr(v_{ij}))$.

Exemple. La sémantique probabiliste PS_3 du services s_3 (Figure3.4) est une *Sémantique Probabiliste non-exhaustive*, parce que la somme des probabilités de toutes ses vues sémantiques possibles est égale à 0,6 (inferieur à 1).

Contrairement à la *Sémantique Probabiliste non-exhaustive*, Dans le cas de la *Sémantique Probabiliste exhaustive*, la somme des probabilités des vues sémantiques possibles est égale à 1. Autrement dit, les vues sémantiques possibles sont uniquement celles qui sont décrites dans l'ensemble $psv(s_i)$, i.e., l'annotateur arrive à définir toutes le vues sémantiques qui peuvent décrire l'interprétation du service s_i.

Définition 3.5 (***Sémantique Probabiliste exhaustive***) : Soit $S = \{s_1, s_2, ..., s_m\}$ un ensemble de services Daas, et soit MO une ontologie de domaine. Soit $PS_i = (psv(s_i), Pr)$ une *Sémantique Probabiliste* pour $s_i \in S$ (formulée à travers MO). PS_i est une *Sémantique Probabiliste exhaustive* $\Longleftrightarrow \sum_{v_{ij} \in psv(s_i)} Pr(v_{ij}) = 1$.

Exemple. Les sémantiques probabilistes PS_1, PS_2 et PS_4 sont des *Sémantiques Probabilistes exhaustives*, et la somme des probabilités de leurs vues sémantiques est égale à 1.

3.2 Génération des sémantiques probabilistes

L'approche que nous proposons pour la modélisation et la composition des services dont la sémantique est incertaine est complètement indépendante des méthodes qui peuvent être utilisées pour générer les sémantiques probabilistes. Les fournisseurs de services sont libres d'utiliser n'importe quelle méthode pour découvrir les vues sémantiques possibles de leurs services ainsi que leurs probabilités.

Une méthode possible pour générer les vues sémantiques d'un service est d'exploiter les techniques automatiques de *Schema Matching* [31], où l'ontologie joue le rôle du schéma médiateur, et les services DaaS sont considérés comme étant des sources de données locales. La première étape pour générer les vues sémantiques possibles d'un service consiste à calculer les *Correspondances Pondérées* entre les entrées/sorties du service [3] et les concepts ontologiques (i.e., plus précisément les Data-Type Properties).

Les correspondances pondérées sont créées en se basant sur un ensemble de méthodes de *Schema Matching*. Chaque méthode prend en paramètre un service $s_k(a_1,...,a_n)$, et un ensemble de Data-Type Properties $DT(dt_1,...,dt_m)$ (à partir d'une ontologie), et retourne une matrice de similarité $M(s_k, DT)$ obtenue par l'application d'une technique ou d'une mesure de similarité. Chaque $m_{ij}(a_i, dt_j) \in M$ représente le degré de similarité entre le i-ème paramètre de s_k et la j-ème Data-Type Property dans DT. $m_{ij}(a_i, dt_j)$ est un nombre réel qui appartient à l'intervalle [0, 1]. Plus ce nombre est élevé, plus il est probable que la correspondance $a_i \approx dt_j$ soit correcte. Le système de matching peut employer diverses mesures de similarité menant à différentes matrices. Nous discutons ci-dessous certaines de ces mesures de similarité.

Matching à base de nom : Les vues possibles d'un service DaaS peuvent différer sur la manière dont leurs paramètres sont interprétés. Certaines méthodes sont basées sur des techniques de matching textuelle (e.g., TF/IDF, Jaro, etc), où le degré de similarité entre un paramètre de service a_i et une Data-Type Property dt_j est calculé en fonction de leurs noms (nous comparons aussi les termes des concepts ontologiques associés).

Matching à base de contenu : Les vues possibles d'un service DaaS peuvent également différer sur la manière dont les données retournées sont interprétées. Différentes techniques peuvent être utilisées pour calculer les degrés de similarité en se basant sur les données encapsulées par un service, e.g., la superposition des valeurs, les classificateurs, etc.

Afin de calculer les correspondances pondérées finales, nous devons combiner les résultats obtenus par les différentes méthodes de matching utilisées, i.e., les différentes matrices de similarité devraient être fusionnées. Pour cette raison, différentes fonctions de fusion peuvent être appliquées, e.g., *average*, *minimum*, ou *maximum*, etc. Par exemple, si l

3. Nous utilisons les termes " entrée/sortie de service" et "paramètres de service" de manière interchangeable.

différentes méthodes ont été utilisées pour calculer la correspondance pondérée wc_{ij} entre le paramètre de service a_i et la Data-Type Property dt_j, alors la fonction *average* calcule la correspondance pondérée finale entre ces deux éléments de la façon suivante :

$$wc_{ij}(a_i,\, dt_j) = \left[\sum_{x=1}^{l} match(x, a_i, dt_j)\right] / l$$

où $match(x, a_i, dt_j)$ est le degré de similarité m_{ij} entre a_i et dt_j, et est produit par la x-ème methode.

Chaque paramètre de service a_i sera associé à un ensemble de Data-Type Properties possibles à travers un ensemble de correspondances pondérées WC_i=$\{wc_{i1},...,wc_{im}\}$. Cet ensemble est raffiné en ne gardant que les correspondances pondérées qui sont supérieures ou égales à un seuil fixé par un expert de domaine.

Dans l'étape suivante, nous générons l'ensemble des vues sémantiques possibles par la combinaison des correspondances pondérées possibles associées aux paramètres du service. Chaque vue sémantique $v_{kh} \in psv(s_k)$ est obtenue en associant à chaque paramètre de service a_i une de ses correspondances pondérées possibles $wc_{ij} \in WC_i$. Les vues sémantiques possibles dans $psv(s_k) = \{v_{k1}, v_{k2}, ..., v_{kn}\}$ sont associées avec les probabilités $\{p_{k1}, p_{k2}, ..., p_{kn}\}$, qui sont calculées en résolvant les contraintes suivantes (OPT) [27] :

`maximize` $\sum_{h=1}^{n} -p_{kh} * \log p_{kh}$ tel que :

- $\forall h \in [1, n]$, $0 \leq p_{kh} \leq 1$,
- $\sum_{h=1}^{n} p_{kh} = 1$
- $\forall\ a_i \in s_k : \sum_{h=1}^{n} p_{kh} = wc_{ij}$, s.t. a_i correspond à la data-type property dt_j dans v_{kh}

.

Ces contraintes ont été proposées dans [27] pour calculer les probabilités des mappings possibles dans un système d'intégration de données de type *LAV* (Local As View). Dans la dernière étape, l'annotateur complète la génération des sémantiques probabilistes en créant manuellement les vues sémantiques finales (i.e., requête SPARQL) en se basant sur les correspondances associées aux entrées/sorties du service.

3.3 Registre de services probabiliste

Un service DaaS est rarement utilisé seul, i.e., il est souvent combiné (i.e., composé) avec d'autres services pour répondre à des requêtes utilisateurs complexes. Cependant, le

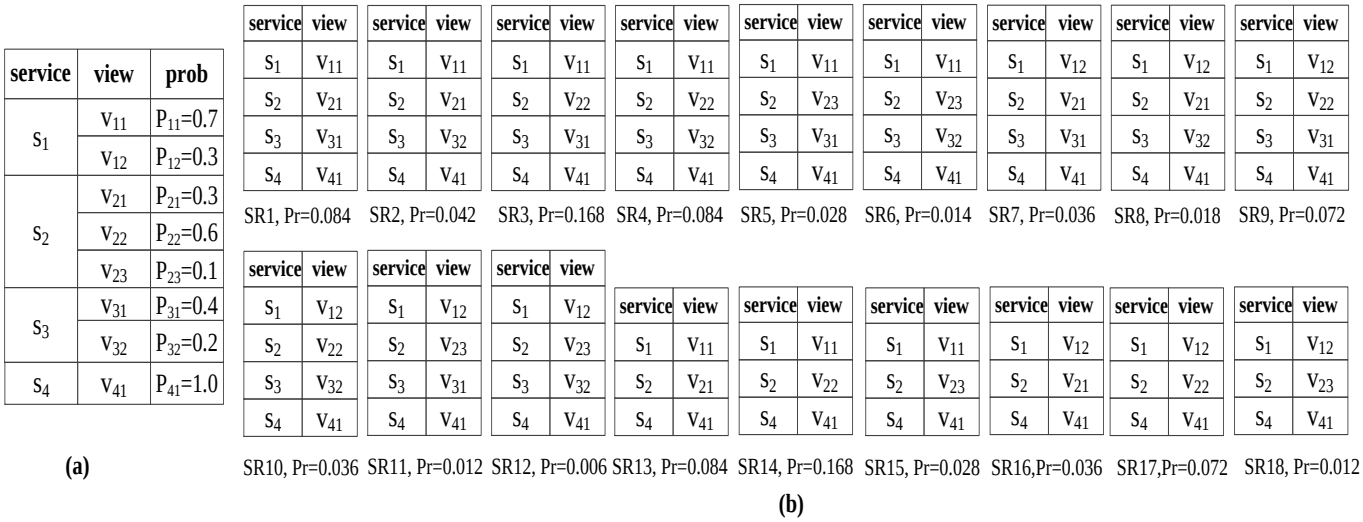
modèle *Sémantique Probabiliste* présenté précédemment ne prend en considération qu'un service à la fois, tout en énumérant ses vues sémantiques possibles avec leurs probabilités. Pour répondre à ce problème, et ainsi fournir un formalisme de représentation plus riche et plus générique, nous proposons d'adopter le concept des *Bases de Données Probabilistes*[4] [11, 25, 114] pour modéliser un ensemble de services avec leurs sémantiques incertaines.

Un ensemble de services DaaS $S = \{s_1, s_2, ..., s_m\}$ à sémantique incertaine est défini par une table probabiliste dont le schéma est : <*service*, *view*, *prob*>, ce que nous appelons *Registe de services probabiliste* et nous le notons par PSR. Les attributs *service* et *view* représentent respectivement l'identifiant et l'une des vues sémantiques possibles de chaque service dans l'ensemble S. Chaque couple $<s_i, v_{ij}>$ ($s_i \in S$, $v_{ij} \in psv(s_i)$) est représenté par un tuple t dans PSR, tel que la probabilité de t est égale à $Pr(v_{ij})$. Chaque tuple $t :<s_i, v_{ij}, Pr(v_{ij})> \in PSR$ signifie que la vue sémantique v_{ij} décrit l'interprétation du service s_i avec une probabilité qui est égale à $Pr(v_{ij})$.

Dans le registre de service probabiliste PSR, Les différents tuples qui décrivent la sémantique d'un même service sont considérés comme étant des *événements probabiliste disjoints*. Ceci s'explique par le fait qu'une seule vue sémantique possible est correcte. En revanche, les tuples qui décrivent la sémantique des services différents sont considérés comme étant *événements probabilistes indépendants* (nous supposons que les sources de données accédées par les différents services sont indépendantes). Dans la littérature, différents modèles et formalismes ont été proposés pour tenir compte des corrélations (i.e., dépendances probabilistes) qui peuvent exister entre les tuples d'une base de données probabiliste [5, 11, 108]. Parmi ces modèles, le modèle *Bloc-Indépendant-Disjoint* (noté BID) [24] est le plus adéquat pour représenter les types de corrélation qui existent entre les tuples (ou les vues sémantiques) du PSR (i.e., la disjonction et l'indépendance). Dans un tel modèle, les tuples dans un registre de services probabiliste sont partitionnés en un ensemble de blocs, où les tuples du même bloc sont des évènements probabilistes disjoints, alors que les tuples qui appartiennent à des blocs différents sont indépendants.

Exemple. La figure 3.5 (a) montre le registre de services probabiliste PSR_1 qui correspond à notre exemple de motivation. PSR_1 contient quatre blocs, un bloc par service ; les

4. Les bases de données probabilistes sont utilisées comme étant un formalisme et non pas pour représenter les données incertaines



service	view	prob
s_1	v_{11}	P_{11}=0.7
	v_{12}	P_{12}=0.3
s_2	v_{21}	P_{21}=0.3
	v_{22}	P_{22}=0.6
	v_{23}	P_{23}=0.1
s_3	v_{31}	P_{31}=0.4
	v_{32}	P_{32}=0.2
s_4	v_{41}	P_{41}=1.0

(a)

service	view
s_1	v_{11}
s_2	v_{21}
s_3	v_{31}
s_4	v_{41}

SR1, Pr=0.084

service	view
s_1	v_{11}
s_2	v_{21}
s_3	v_{32}
s_4	v_{41}

SR2, Pr=0.042

service	view
s_1	v_{11}
s_2	v_{22}
s_3	v_{31}
s_4	v_{41}

SR3, Pr=0.168

service	view
s_1	v_{11}
s_2	v_{22}
s_3	v_{32}
s_4	v_{41}

SR4, Pr=0.084

service	view
s_1	v_{11}
s_2	v_{23}
s_3	v_{31}
s_4	v_{41}

SR5, Pr=0.028

service	view
s_1	v_{11}
s_2	v_{23}
s_3	v_{32}
s_4	v_{41}

SR6, Pr=0.014

service	view
s_1	v_{12}
s_2	v_{21}
s_3	v_{31}
s_4	v_{41}

SR7, Pr=0.036

service	view
s_1	v_{12}
s_2	v_{21}
s_3	v_{32}
s_4	v_{41}

SR8, Pr=0.018

service	view
s_1	v_{12}
s_2	v_{22}
s_3	v_{31}
s_4	v_{41}

SR9, Pr=0.072

service	view
s_1	v_{12}
s_2	v_{22}
s_3	v_{32}
s_4	v_{41}

SR10, Pr=0.036

service	view
s_1	v_{12}
s_2	v_{23}
s_3	v_{31}
s_4	v_{41}

SR11, Pr=0.012

service	view
s_1	v_{12}
s_2	v_{23}
s_3	v_{32}
s_4	v_{41}

SR12, Pr=0.006

service	view
s_1	v_{11}
s_2	v_{21}
s_4	v_{41}

SR13, Pr=0.084

service	view
s_1	v_{11}
s_2	v_{22}
s_4	v_{41}

SR14, Pr=0.168

service	view
s_1	v_{11}
s_2	v_{23}
s_4	v_{41}

SR15, Pr=0.028

service	view
s_1	v_{12}
s_2	v_{21}
s_4	v_{41}

SR16,Pr=0.036

service	view
s_1	v_{12}
s_2	v_{22}
s_4	v_{41}

SR17,Pr=0.072

service	view
s_1	v_{12}
s_2	v_{23}
s_4	v_{41}

SR18, Pr=0.012

(b)

FIGURE 3.5 – (a) Registre de services probabiliste PSR_1 avec la représentation BID ; (b) Registres de services possibles de PSR_1.

vues sémantiques dans chaque bloc sont vues comme étant des événements probabilistes disjoints (e.g., v_{11}, v_{12} sont disjointes, de même pour v_{31}, v_{32}, etc), alors que les vues sémantiques de blocs différents sont indépendantes, e.g., v_{11}, v_{21}, v_{41}, etc.

Formellement le registre de services probabiliste PSR est défini comme suit.

Définition 3.6 (***PSR avec le modèle BID***) : un registre de services probabiliste PSR :<$service$, $view$, $prob$> est une table probabiliste BID, si pour chaque deux tuples t_k, $t_h \in PSR$:

- $t_k.service$=$t_h.service$ $\rightarrow$ [$t_k.view$, $t_h.view$ sont disjoints] $\wedge$ [$Pr(t_k, t_h) = 0$]
- $t_k.service \neq t_h.service \rightarrow$[$t_k.view$, $t_h.view$ sont independants] $\wedge$ [$Pr(t_k, t_h) = t_k.prob \times t_h.prob$]

La sémantique de PSR peut être définie en se basant sur la *Sémantique des mondes possibles* [25], où PSR est considéré comme étant une distribution de probabilité sur un ensemble de registres de services déterministes $\{SR_1, SR_2, ..., SR_n\}$, ce que nous appelons *registres de services possibles* (noté par $pwd(SR)$). $pwd(SR)$ est construit en générant les différentes combinaisons possibles des tuples dans PSR. Chaque combinaison doit inclure au maximum un tuple de chaque bloc, puisque les tuples d'un même bloc sont disjoints.

Chaque registre de service possible SR_k ne contient que des tuple certains (déterministes), i.e., les services dans chaque registre possible ont une sémantique certaine (associés à l'une de leurs vues sémantiques possibles).

La ***probabilité jointe*** de chaque registre possible SR_k est calculée en se basant sur les probabilités des tuples impliqués. Plus précisément, la probabilité de SR_k est un produit avec un seul facteur pour chaque service s_i (i.e., pour chaque bloc) dans PSR, tel que si s_i figure dans SR_k avec une vue sémantique v_{ij}, alors son facteur est égale à $Pr(v_{ij})$; sinon il est égale à $[1 - \sum_{v_{ij} \in psv(s_i)} Pr(v_{ij})]$ (i.e., s_i ne figure pas dans le registre possible SR_k).

Définition 3.7 (***Sémantique du PSR***) : La sémantique d'un registre de services probabiliste est une distribution de probabilité sur l'ensemble des registres possibles $pwd(SR)$. Plus précisément, PSR est un espace probabiliste $PSR = (pwd(SR), Pr)$, où $pwd(SR) = \{SR_1, SR_2, ..., SR_n\}$, et Pr est une fonction de probabilité $Pr : pwd(SR) \rightarrow]0, 1]$ tel que :

- Pour chaque $SR_k \in pwd(SR)$, $Pr(SR_k) = \prod_{<s_i, v_{ix}> \in SR_k} Pr(v_{ix}) \times \prod_{s_i \notin SR_k} 1 - [\sum_{v_{ij} \in psv(s_i)} Pr(v_{ij})]$.
- $\sum_{SR_k \in pwd(SR)} Pr(SR_k) = 1$.
- Pour tous $k, h \in [1, n], k \neq h \rightarrow Pr(SR_k, SR_h) = 0$ (i.e., tous les registres de services possibles sont disjoints).

Le nombre des registres de services possibles est exponentiel. Si le nombre de tuples dans PSR est égale à n, alors $|pwd(SR)|$ peut aller jusqu'à 2^n.

Exemple. La figure 3.5 (b) montre les dix-huit registres de services possibles correspondants à PSR_1 dans la Figure 3.5 (a). SR_1 implique tous les quatre services comme suit : s_1 est associé avec v_{11}, s_2 est associé avec v_{21}, s_3 est associé avec v_{31}, et s_4 avec v_{41}. Sa probabilité est calculée comme suit : $Pr(SR_1) = Pr(v_{11}) * Pr(v_{21}) * Pr(v_{31}) * Pr(v_{41}) = 0.7 * 0.3 * 0.4 * 1.0 = 0.084$. SR_{18} ne contient aucune vue sémantique pour s_3 (i.e., s_3 n'apparaît pas dans SR_{18}), sa probabilité est calculée comme suit : $Pr(SR_{18}) = Pr(v_{12}) * Pr(v_{23}) * (1 - Pr(v_{31}) - Pr(v_{32})) * Pr(v_{41}) = 0.3 * 0.1 *$ (1- 0.4 - 0.2) $* 1.0 = 0.012$.

Dans la figure 3.5 (b), les services s_1, s_2, s_4 figurent dans tous les registres de services possibles. Ceci s'explique par le fait que les sémantiques probabilistes associées à ces services sont *exhaustives* (i.e., la somme des probabilités des vues sémantiques possibles est égale à 1), et par conséquent la probabilité des registres qui n'impliquent pas ces services

est égale à 0 (i.e., ils sont ignorés). En revanche, la sémantique probabiliste du service s_3 n'est pas exhaustive (i.e., il peut y avoir d'autres vues sémantiques), ce qui permet d'obtenir des registres de services (e.g., $SR_{13}, SR_{14}, SR_{15}, SR_{16}, SR_{17}, SR_{18}$) qui n'impliquent pas s_3, et dont la probabilité est supérieure à 0. Dans ces registres de services s_3 est associé à une sémantique inconnue.

4 Interprétation probabiliste de compositions de services

Dans la section précédente, nous avons proposé un modèle probabiliste pour les services incertains. Dans cette section, nous nous intéressons au problème suivant : soit une composition impliquant des services dont la sémantique est incertaine, alors dans ce cas, quelles sont les interprétations possibles de cette composition ?, et quelles sont leurs probabilités ?.

Nous définissons d'abord l'interprétation d'une composition certaine (i.e., impliquant des services déterministes), puis nous abordons le cas incertain, tout en basant sur le concept de *Registre de services probabiliste*.

4.1 Interprétation des compositions certaines

Une composition décrit l'interaction entre deux ou plusieurs services afin de répondre à une requête complexe qui ne peut pas être résolue par un seul service. L'interprétation d'une composition (i.e., sa sémantique) doit être définie explicitement pour permettre sa réutilisation (comme étant un nouveau service). L'interprétation d'une composition décrit la sémantique des données retournées par l'exécution de cette dernière.

Une composition de services est caractérisée par deux facteurs : les services impliqués, et le plan de composition qui décrit l'orchestration des services composés. Informellement, un plan de composition est un graphe orienté acyclique [112], dans lequel les nœuds représentent les services composés et les opérateurs de traitement de données (e.g., sélection, projection, jointure, etc). Les arcs représentent les échanges de données faits entre ces nœuds. Deux compositions qui impliquent les mêmes services n'ont pas nécessairement le même plan de composition. Cela est dû aux différents opérateurs algébriques qui sont impliqués dans les plans des compositions.

Formellement, une composition est définie comme suit.

Définition 3.8 (***Composition de services***) : Soit SR un registre de service, une composition C_i est un couple $<S_{C_i}, \rho_{C_i}>$, où :

– $S_{C_i} \subseteq SR$, est l'ensemble de services impliqués dans C_i.

– ρ_{C_i} est le plan de composition de C_i. ρ_{C_i} est un graphe orienté acyclique représenté par un couple $<N, E>$, où N représente les nœuds. Chaque nœud peut être soit un service soit un opérateur algébrique $N \in \{S_{C_i}, \Pi, \sigma, \bowtie\}$, et E représente les arcs du graphe, i.e., les données échangées entre les nœuds.

Exemple. Soit $s_a(\$x, ?y, ?f)$, $s_b(\$d, ?w)$ et $s_c(\$e, ?t)$ des services DaaS. La figure 3.6 donne un exemple de composition de ces services (i.e., le plan de composition ρ_C). Nous supposons que dans un plan de composition, les entrées d'un nœud de service sont toujours combinées (e.g., par un moteur de composition) avec ses sorties. Cela nous permet de calculer le résultat final d'une composition en joignant les sorties des services qui sont dans les nœuds feuilles de ρ_{C_i} (i.e., s_b et s_c).

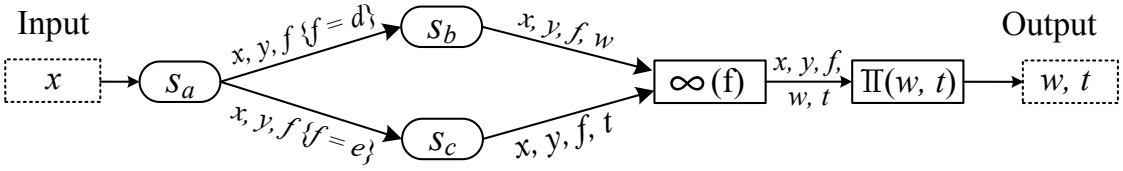


FIGURE 3.6 – Le plan de composition de C_i.

Intuitivement, l'interprétation (i.e., la sémantique) d'une composition peut être représentée de façon déclarative par une vue sémantique (i.e., une requête à travers une ontologie médiatrice). Cela s'explique par le fait qu'une composition peut également être considérée comme étant un nouveau service. Cette vue sémantique est construite en combinant les vues sémantiques des services composés, et en respectant les différents opérateurs algébriques spécifiés dans le plan de composition.

Définition 3.9 (***Interprétation d'une composition certaine***) : Soit $S_{C_i} = \{s_1, ..., s_n\}$ l'ensemble de services impliqués dans une composition C_i, et soit $V = \{v_1, ..., v_n\}$ leurs vues sémantiques. L'interprétation de C_i, notée par C_{i_intp}, est une vue sémantique formulée à travers V en appliquant ρ_{C_i} comme suit :

– $v_1, ..., v_n$ sont les prédicats qui constituent le corps du C_{i_intp}.

– Les variables d'entrées/sorties de C_i sont les variables distinguées (i.e., paramètres de prédicat de tête) de C_{i_intp}.

– Si ρ_{C_i} mappe une variable de sortie a d'un service s_k avec une variable d'entrée b d'un service s_h, alors a et b sont mappées à la même variable dans C_{i_intp}.

– Si un opérateur de jointure joint deux variables dans ρ_{C_i} alors ces variables sont mappées à la même variable C_{i_intp}.

– Les variables qui sont filtrées par les opérateurs de projection sont des variables existentielles dans C_{i_intp}.

Example. Nous continuons avec l'exemple précédent de la figure 3.6. Supposons que les services $s_a(\$x, ?y, ?f)$, $s_b(\$d, ?w)$ et $s_c(\$e, ?t)$ sont respectivement décrits par les vues sémantiques : $v_a(\$x, ?y, ?f)$, $v_b(\$d, ?w)$ et $v_c(\$e, ?t)$. L'interprétation de C_i (i.e., sa vue sémantique correspondante) peut être exprimée (dans la notation Datalog) à travers ces vues comme suit :

$+C_{i\_intp}$= `V($x,?w,?t):- V`$_a$`($x,?y,?z), V`$_b$`($z,?w), V`$_c$`($z,?t)`

Elle est construite en se basant sur ρ_{C_i}, tel que : (1) les entrées/sorties de C_i deviennent des variables distinguées dans V (i.e., elles apparaissent dans son prédicat de tête) ; (2) Si un paramètre de sortie d'un service s_k est un paramètre d'entrée pour un autre service s_h (i.e., s_k et s_h sont liés par un arc dans ρ_{C_i}), alors ces deux paramètres correspondent à la même variable dans le corps de V. Par exemple, le paramètre f du service s_1 et le paramètre d du service s_2 correspondent à la même variable z dans le corps de V (le même principe s'applique lorsque deux paramètres différents sont liés par un opérateur de jointure).

4.2 Interprétation des compositions incertaines

Nous avons représenté la sémantique incertaine des services DaaS par un ensemble de vues sémantiques possibles, chacune ayant une probabilité. Par conséquent, une composition peut avoir différentes interprétations possibles issues des différentes combinaisons des vues sémantiques de ses services impliqués.

L'interprétation d'une composition incertaine est définie en se basant sur le concept

registre de services probabiliste PSR. Une composition C_i a un ensemble d'interprétations possibles $\{C_{i\_intp\_1}, C_{i\_intp\_2}, ..., C_{i\_intp\_n}\}$ noté par $pwd(C_{i\_intp})$, et qui est obtenu en calculant l'*interprétation de composition certaine* dans chaque registre de services possible de $pwd(SR)$ auquel C_i détient une interprétation (i.e., une composition détient une interprétation dans un registre possible SR_h si l'ensemble de ses services composés appartiennent tous à ce registre). Notons qu'une composition C_i peut avoir au maximum une seule interprétation possible dans chaque registre de service SR_h. La probabilité de chaque interprétation possible $C_{i\_intp\_k}$ est égale à la somme des probabilités des registres de services possibles dans $pwd(SR)$ qui retournent $C_{i\_intp\_k}$.

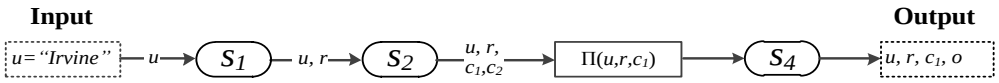
Définition 3.10 (***Interprétation d'une composition incertaine***) : Soit $PSR = (pwd(SR), Pr)$ un registre de services probabiliste qui définit une distribution de probabilité sur un ensemble de registres de services possibles $pwd(SR) = \{SR_1, ..., SR_n\}$, et soit C_i une composition de services. C_i accepte différentes interprétations possibles $pwd(C_{i\_intp}) = \{C_{i\_intp\_1}, ..., C_{i\_intp\_p}\}$, tel que la probabilité de chacune est calculée comme suit :

$$Pr(C_{i\_intp\_k}) = \sum Pr(SR_h)$$

où $SR_h \in pwd(SR)$, et SR_h retourne $C_{i\_intp\_k}$.

Example. Soit C_1 une composition qui implique les services s_1, s_2 et s_4 de notre exemple de motivation (voir la figure 3.7 (a)). Comme il est mentionné précédemment, les sémantiques de ces services sont incertaines, et sont représentées dans le registre de services probabiliste PSR_1 dans la figure 3.5(a). Les interprétations de C_1 peuvent être calculées en appliquant C_1 sur chacun des dix-huit registres de services possibles correspondants à PSR_1 (voir la figure 3.5(b)), et ensuite en vérifiant si elle détient une interprétation ou non. Cela donne six interprétations possibles décrites dans la figure 3.7(b). L'interprétation $C_{1\_intp\_1}$ est obtenue à partir des registres SR_1, SR_2, SR_{13}, et par conséquent sa probabilité est la somme des probabilités de ces registres : $Pr(C_{1\_intp\_1}) = Pr(SR_1) + Pr(SR_2) + Pr(SR_{13}) = 0.21$. Le même principe est appliqué pour calculer les probabilités des cinq autres interprétations. Chacune de ces six interprétations donne une sémantique spécifique aux données retournées par la composition C_1 (voir la figure 3.7(c)). Par exemple, si la composition C_1 retourne le tuple <"*Bob*","*0862148*","*Orange*" >, alors selon $C_{1\_intp\_1}$, *Bob* est l'un des

(a)



(b)

Interprétations possibles	Probabilité	description
$C_{1\_intp\_1}$($u, ?r, ?$c_1$, ?o):-$v_{11}$($u, ?r) v_{21}($r, ?$c_1$,?$c_2$) v_{41}(c_1, ?o)	$Pr(SR_1)+Pr(SR_2)+Pr(SR_{13})$=0.21	Retourne les téléphones mobiles (et leurs opérateurs) des chercheurs actuels de l'université d'irvine
$C_{1\_intp\_2}$($u, ?r, ?$c_1$, ?o):-$v_{11}$($u, ?r) v_{22}($r, ?$c_1$,?$c_2$) v_{41}(c_1, ?o)	$Pr(SR_3)+Pr(SR_4)+Pr(SR_{14})$=0.42	Retourne les téléphones domiciles (et leurs opérateurs) des chercheurs actuels de l'université d'irvine
$C_{1\_intp\_3}$($u, ?r, ?$c_1$, ?o):-$v_{11}$($u, ?r) v_{23}($r, ?$c_1$,?$c_2$) v_{41}(c_1, ?o)	$Pr(SR_5)+Pr(SR_6)+ Pr(SR_{15})$=0.07	Retourne les numéros de fax (et leurs opérateurs) des chercheurs actuels de l'université d'irvine
$C_{1\_intp\_4}$($u, ?r, ?$c_1$, ?o):-$v_{12}$($u, ?r) v_{21}($r, ?$c_1$,?$c_2$) v_{41}(c_1, ?o)	$Pr(SR_7)+Pr(SR_8)+ Pr(SR_{16})$=0.09	Retourne les téléphones mobiles (et leurs opérateurs) des anciens chercheurs de l'université d'irvine
$C_{1\_intp\_5}$($u, ?r, ?$c_1$, ?o):-$v_{12}$($u, ?r) v_{22}($r, ?$c_1$,?$c_2$) v_{41}(c_1, ?o)	$Pr(SR_9)+Pr(SR_{10})+ Pr(SR_{17})$=0.18	Retourne les téléphones domiciles (et leurs opérateurs) des anciens chercheurs de l'université d'irvine
$C_{1\_intp\_6}$($u, ?r, ?$c_1$, ?o):-$v_{12}$($u, ?r) v_{23}($r, ?$c_1$,?$c_2$) v_{41}(c_1, ?o)	$Pr(SR_{11})+Pr(SR_{12})+Pr(SR_{18})$=0.03	Retourne les numéros de fax (et leurs opérateurs) des anciens chercheurs de l'université d'irvine

(c)

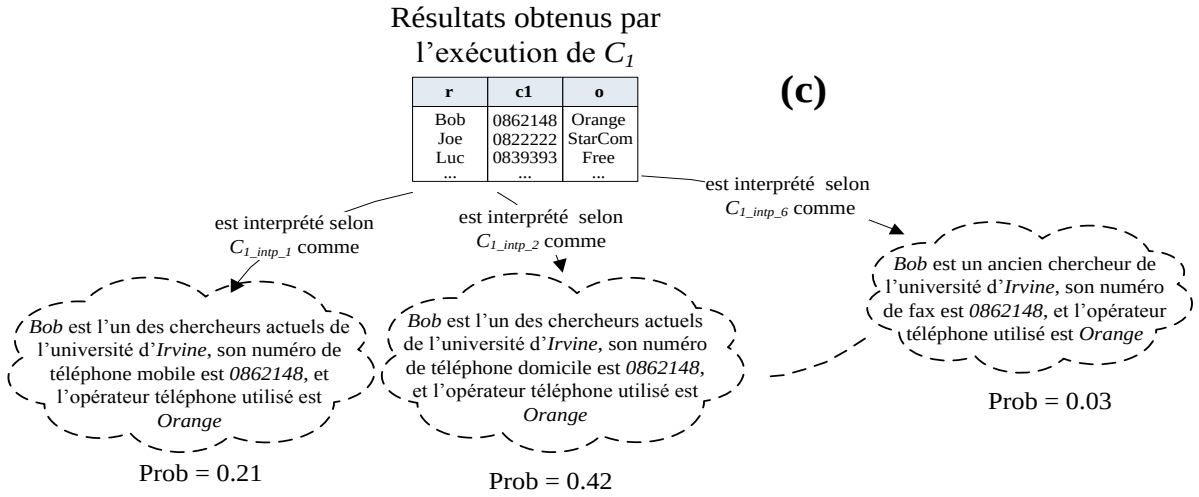


FIGURE 3.7 – (a) Composition C_1 ; (b) Interprétations possibles de C_1 ; (c) Les interprétations des données retournées par la composition C_1.

chercheurs *actuels* à l'université d'Irvine, son numéro de téléphone mobile est 086214895, et l'opérateur téléphonique utilisé est *Orange* (cette interprétation a une probabilité 0.21) ; tandis que selon $C_{1\_intp\_6}$ *Bob* est un *ancien* chercheur de l'université d'Irvine, 086214895 est son numéro de fax, et *Orange* est l'opérateur de fax (cette interprétation a une probabilité de 0.03).

Il est intéressant de mentionner que l'exécution d'une composition retournera toujours les mêmes résultats, indépendamment de l'interprétation adoptée. Cela est dû au fait que l'incertitude concerne l'interprétation des données retournées, et non pas les données elles-mêmes, i.e., les valeurs des données sont certaines.

Générer les interprétations possibles d'une composition via l'ensemble des registres de services possibles est intractable, puisque le nombre de registres de services possibles est exponentiel. Nous proposons un théorème qui nous permet de calculer efficacement la probabilité d'une interprétation existante (i.e., dans un temps polynomial) sans avoir besoin de matérialiser les registres de services possibles. Le théorème calcule la probabilité d'une interprétation possible en multipliant les probabilités des vues sémantiques impliquées.

Théorèm 1. *Soit $C_{i\_intp\_k}$ une interprétation possible d'une composition C_i. La probabilité de $C_{i\_intp\_k}$ est égale au produit des probabilités des vues sémantiques impliquées, tel que :* $Pr(C_{i\_intp\_k}) = \prod_{v_{jt} \in C_{i\_intp\_k}} Pr(v_{jt})$.

Démonstration : Soit $C_{i\_intp\_k}$ une interprétation possible d'une composition C_i, et soit $PSR = (pwd(SR), Pr)$ un registre de services probabiliste représenté par le modèle BID. Chaque registre de services possible $SR_h \in pwd(SR)$ contient des vues sémantiques appartenant à des blocs différents dans PSR (i.e., toutes les vues sémantiques dans chaque registre de services possible sont des événements probabilistes indépendants). Selon la ***Définition 4.3*** (*Interprétation d'une composition incertaine*), la probabilité d'une interprétation possible $C_{i\_intp\_k}$ est $\sum Pr(SR_h)$, où SR_h retourne $C_{i\_intp\_k}$, i.e., $C_{i\_intp\_k}$ comme tout registre de services possible, elle contient également des vues sémantiques qui sont considérées comme étant des événements probabilistes indépendants. Dans la théorie des probabilités, si un ensemble d'événements probabilistes $X = \{x_1, x_2, ..., x_n\}$ est indépendant, alors la probabilité $Pr(x_1, x_2, ..., x_n) = \prod_{i=1}^{n} Pr(x_i)$. Puisque l'interprétation $C_{i\_intp\_k}$ contient des vues qui sont indépendantes, par conséquent la probabilité de $C_{i\_intp\_k}$ est égale au produit des probabilités des vues sémantiques impliquées, i.e., $Pr(C_{i\_intp\_k}) = \prod_{v_{jt} \in C_{i\_intp\_k}} Pr(v_{jt})$.

Example. L'interprétation possible $C_{1\_intp\_1}$ dans la figure 3.7(b) est obtenue à partir des registres possibles SR_1, SR_2, SR_{13}, par conséquent $Pr(C_{1\_intp\_1}) = Pr(SR_1) + Pr(SR_2) + Pr(SR_{13}) = 0.21$. En se basant sur le **Théorème 1**, nous pouvons obtenir la même probabilité simplement en multipliant les probabilités des vues sémantiques impliquées dans $C_{1\_intp\_1}$: $Pr(C_{1\_intp\_1}) = Pr(v_{11}) * Pr(v_{21}) * Pr(v_{41}) = 0.7 * 0.3 * 1.0 = 0.21$.

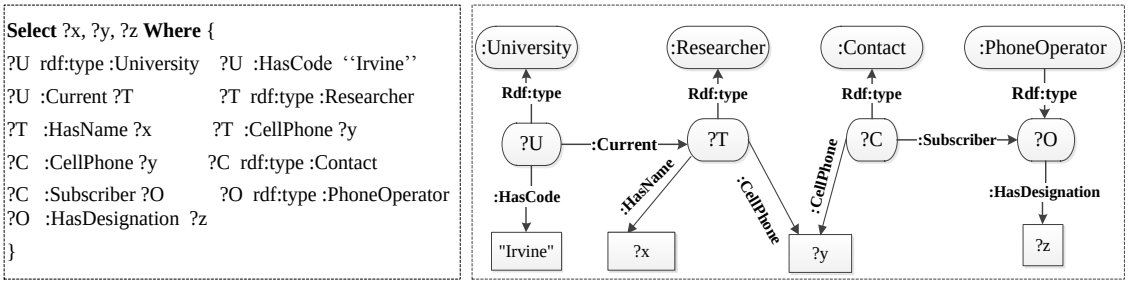


FIGURE 3.8 – La requête utilisateur Q_1.

5 Evaluation de requêtes à travers le registre PSR

Dans la section précédente, nous avons montré comment calculer les différentes interprétations possibles (avec leurs probabilités) d'une composition existante. Dans cette section, nous traitons le problème de la réécriture des requêtes dans le contexte des services DaaS incertains. Soit Q une requête utilisateur formulée à travers une ontologie médiatrice, et soit PSR un registre de services probabiliste qui décrit les sémantiques probabilistes d'un ensemble de services DaaS. Comment obtenir l'ensemble des compositions qui répondent aux besoins de la requête Q ?, et comment calculer leurs probabilités ?.

Dans cette section, notre objectif n'est pas de proposer un nouvel algorithme de réécriture de requêtes à base de services DaaS. Au lieu de cela, nous utilisons un algorithme de réécriture existant [9] qui nous permet de découvrir, sélectionner et de composer les services dont les vues sémantiques sont pertinentes à la requête. Ensuite, nous nous concentrons sur le calcul des probabilités des compositions en prenant en considération l'incertitude des services impliqués.

Une composition C_i est une solution pour une requête utilisateur Q, si et seulement si, $\exists\ C_{i\_intp\_x} \in pwd(C_{i\_intp})$ de C_i, tel que $C_{i\_intp\_x}$ couvre Q (i.e., au moins l'une des interprétations possibles de la composition C_i couvre Q).

Tout au long de cette section, nous utilisons la requête Q_1 de notre exemple de motivation. La figure 3.8 montre la requête SPARQL qui correspond à Q_1, ainsi que sa représentation graphique.

Nous définissons maintenant la sémantique de l'évaluation d'une requête à travers un registre de services probabiliste, i.e., la façon dont une requête est réécrite à travers un registre de services probabiliste. Soit PSR un registre de services probabiliste décrit par

le modèle BID, et soit Q une requête utilisateur. Réécrire Q à travers PSR revient à la réécrire sur chaque registre de services. La réécriture de Q à travers chaque registre de services possible $SR_k \in pwd(SR)$ est réalisée comme dans le cas certain (en utilisant un algorithme de réécriture certain [9]), puisque chaque registre SR_k est déterministe (i.e., chaque service est associé à une est une seule vue sémantique). Le résultat obtenu peut être interprété de deux façons différentes : *Sémantique des Résultats possibles*, et *Sémantique des Compositions possibles.*

5.1 Sémantique des Résultats possibles

Dans le cas des *Résultats possibles*, Q est évaluée sur chaque registre de services possible, ce qui permet de retourner un ensemble de résultats possibles $\{Answer_1, Answer_2,..., Answer_m\}$ noté par $Q^{pwd}(PSR)$. Chaque résultat possible $Answer_k$ dans $Q^{pwd}(PSR)$ contient 0 ou plusieurs compositions (i.e., le résultat $\emptyset$ est inclus). Chaque composition obtenue est associée à une des ses interprétations possibles, déterminée par le registre de service correspondant. En outre, la probabilité d'une composition est égale à la probabilité du résultat possible dont elle fait partie. La probabilité de chaque résultat possible est égale à la somme des probabilités des registres de services possibles qui retournent ce résultat.

Définition 3.11 (***Sémantique des résultats possibles***) : Soit $PSR = (pwd(SR), Pr)$ un registre de services probabiliste qui définit une distribution de probabilité sur un ensemble de registres de services possibles $pwd(SR) = \{SR_1, ..., SR_n\}$, et soit Q une requête utilisateur. Dans la *sémantique des résultats possibles*, le résultat de l'évaluation de Q sur PSR est interprété comme étant une *distribution de probabilité* sur un ensemble de résultats possibles $Q^{pwd}(PSR) = \{Answer_1, Answer_2, ..., Answer_m\}$ obtenu en appliquant Q sur chaque registre de services possible, tel que :

- $Pr(Answer_k) = \sum_{Q(SR_h)=Answer_k} Pr(SR_h)$, où $SR_h \in pwd(SR)$.
- $\sum_{Answer_k \in Q^{pwd}(PSR)} Pr(Answer_k) = 1$.

Example. Nous montrons maintenant la *sémantique des résultats possibles* en utilisant les services (s_1, s_2, s_3, s_4), et la requête Q_1 décrits dans l'exemple de motivation. Les

(a)

Possible Answers	Probability
Answer1={C1,C3,C4}	***Pr***(SR2)=0.042
Answer2={C2,C3,C5}	***Pr***(SR4)=0.084
Answer3={C1}	***Pr***(SR1)+***Pr***(SR13)=0.168
Answer4={C2}	***Pr***(SR3)+***Pr***(SR14)=0.336
Answer5={C3}	***Pr***(SR6)=0.014
Answer6=∅	0.356

(b)

Possible Compositions	Probability
C1	***Pr***(Answer1)+***Pr***(Answer3)=0.21
C2	***Pr***(Answer2)+***Pr***(Answer4)=0.42
C3	***Pr***(Answer1)+***Pr***(Answer2)+***Pr***(Answer5)=0.14
C4	***Pr***(Answer1)=0.042
C5	***Pr***(Answer2)=0.084

(c)

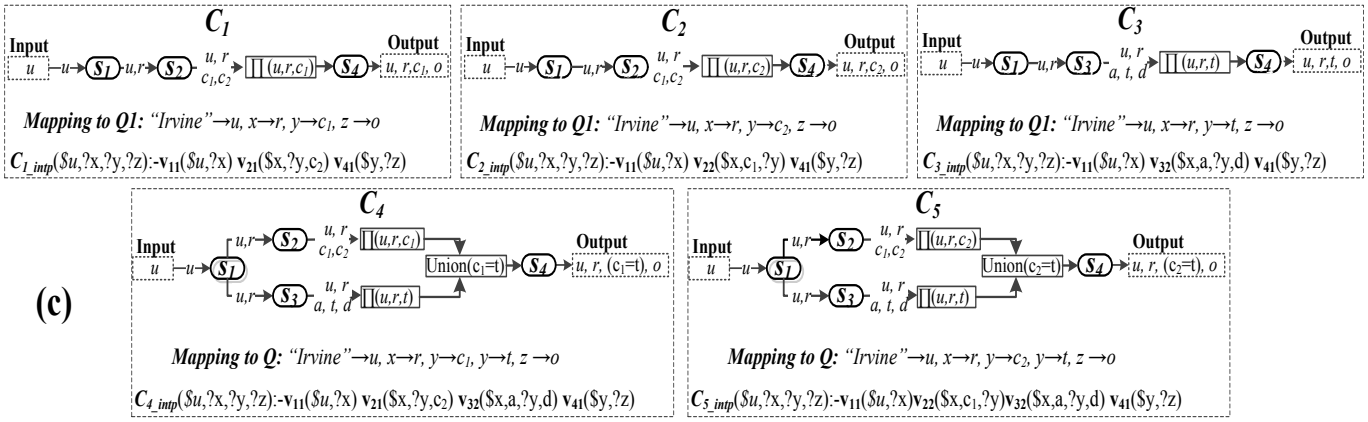


FIGURE 3.9 – (a) Résultats possibles de Q_1. (b) Compositions possibles de Q_1. (c) Compositions obtenues avec leurs interprétations.

sémantiques possibles de ces services sont représentées dans le registre de services probabiliste PSR_1 (voir la figure 3.5(a)). Selon la *sémantique des résultats possibles*, Q_1 accepte les résultats possibles dans la figure 3.9(a). Ces résultats sont obtenus en évaluant Q_1 sur les dix-huit registres de services possibles $\{SR_1, SR_2..., SR_{18}\}$ (voir figure 3.5(a)). Par exemple, l'évaluation de Q_1 sur le registre SR_1 retourne un résultat qui contient une seule composition $Q_1(SR_1) = \{C_1\}$. L'évaluation de Q_1 sur le registre SR_2 retourne un résultat contenant trois compositions $Q_1(SR_2) = \{C_2, C_3, C_4\}$. L'évaluation de Q_1 sur le registre SR_5 retourne un ensemble vide, i.e., $Q_1(SR_5) = \emptyset$. En outre, chaque résultat possible $Answer_k$ est associé à une probabilité qui est égale à la somme des probabilités de tous les registres de service possibles qui retournent $Answer_k$. Dans notre exemple, $Answer_3 = Q_1(SR_1) = Q_1(SR_{13}) = \{C_1\}$. Ainsi, la probabilité de $Answer_3$ est 0.084 + 0.084 = 0.168.

L'ensemble des résultats possibles $Q^{pwd}(PSR)$ peut être très large, et par conséquent il est inintéressant de le retourner à l'utilisateur. En ce sens, nous proposons une autre sémantique appelée *Sémantique des compositions possibles*.

5.2 Sémantique des Compositions possibles

Dans la *sémantique des compositions possibles*, l'évaluation d'une requête Q sur un registre de services probabiliste retourne un ensemble de compositions possibles $Q^{rank}(PSR) = \{C_1, C_2, ..., C_w\}$, tel que chaque composition C_i a une interprétation $C_{i\_intp\_x}$. L'ensemble des compositions possibles $Q^{rank}(PSR)$ est obtenu en faisant l'union de tous les résultats possibles dans $Q^{pwd}(PSR)$. Chaque composition a une probabilité qui est égale à celle de son interprétation. Cette probabilité est obtenue en se basant sur une fonction de probabilité qui calcule pour chaque composition possible C_i une probabilité qui est égale à la somme des probabilités des résultats possibles qui impliquent C_i.

Les compositions dans $Q^{rank}(PSR)$ peuvent être classées en se basant sur leurs probabilités, ce qui permet d'obtenir l'ensemble des meilleures compositions appelé *Top-K* compositions (i.e., la fonction de probabilité classe les compositions possibles).

Définition 3.12 (***Sémantique des compositions possibles***) : Soit $PSR = (pwd(SR), Pr)$ un registre de services probabiliste, soit Q une requête utilisateur, et soit $Q^{pwd}(PSR)$ un ensemble de résultats possibles obtenu en appliquant Q sur PSR. Dans la *sémantique des compositions possibles*, le résultat de l'évaluation de Q sur PSR est interprété comme étant une *fonction de probabilité* sur un ensemble de compositions possibles $Q^{rank}(PSR) = \{C_1, C_2, ..., C_w\}$ obtenu en agrégeant tous les résultats possibles dans $Q^{pwd}(PSR)$. Chaque composition $C_k \in Q^{rank}(PSR)$ a une interprétation $C_{k\_intp\_x}$ dont la probabilité est calculée comme suit :

- $Pr(C_{k\_intp\_x}) = \sum_{<C_k, C_{k\_intp\_x}> \in Answer_h} Pr(Answer_h)$, où $Answer_h \in Q^{pwd}(PSR)$.

Example. Nous utilisons notre exemple pour illustrer la sémantique des compositions possibles. La figure 3.9(b) montre 5 compositions possibles. Chaque composition a une probabilité. Par exemple, la composition C_1 appartient aux résultats $Answer_1$ et $Answer_3$, par conséquent, sa probabilité est égale à $Pr(Answer_1) + Pr(Answer_3)$=0,21. De plus, ces compositions sont classées en se basant sur leurs probabilités.

Nous remarquons bien que la sémantique des compositions possibles est en adéquation avec le **Théorème 1**. Par exemple, la probabilité de la composition C_1 est $Pr(Answer_1) + Pr(Answer_3) = 0.21$, ce qui est égale à celle obtenue par l'utilisation du **Théorème 1**,

i.e., $Pr(v_{11}) * Pr(v_{21}) * Pr(v_{41}) = 0.7 * .0.3 * 1.0 = 0.21$. Ceci s'explique par le fait que la *fonction de probabilité* prend en compte tous les résultats possibles, et par conséquent tous les registres de services possibles qui peuvent retourner une telle composition.

Dans la section suivante, nous proposons un algorithme qui permet de calculer $Q^{rank}(PSR)$ efficacement, i.e., dans un temps optimal.

6 Évaluation efficace de requêtes

L'évaluation des requêtes à travers l'ensemble des registres de services possibles est clairement intractable, puisque le nombre de ces registres est exponentiel. Dans cette section, nous proposons une méthode qui assure l'évaluation efficace des requêtes. Le résultat de notre solution proposée est en adéquation avec la *sémantique des compositions possibles* $Q^{rank}(PSR)$.

Dans notre méthode, une requête Q est appliquée directement sur le PSR, sans avoir besoin de matérialiser l'ensemble des registres de services possibles. A cet effet, PSR est considéré comme étant un registre de services déterministe, où un service avec plusieurs vues sémantiques est traité par l'algorithme de réécriture comme s'il était un ensemble de services (i.e., un service avec n vues sémantiques est considéré par l'algorithme de réécriture comme n services indépendants). Ensuite, dans une étape ultérieure, les compositions dont l'interprétation implique des vues sémantiques disjointes sont ignorées. Enfin, les probabilités des compositions restantes sont calculées en utilisant le **Théorème 1**. Dans ce qui suit, nous détaillons les différentes phases de notre méthode.

6.1 Transformation du registre PSR

Pour réécrire la requête Q directement sur le PSR sans avoir besoin de générer l'ensemble $pwd(SR)$, nous transformons le registre PSR en un seul registre de service déterministe appelé DSR, tel que : chaque bloc indépendant représentant un service s_i et ses différentes vues sémantiques $\{<v_{i1}, Pr(v_{i1})>,<v_{i2}, Pr(v_{i2})>, ...,<v_{im}, Pr(v_{im})>\}$ est transformé en un ensemble de services déterministes $\{<s_{i1}, v_{i1}>,<s_{i2}, v_{i2}>, ...,<s_{im}, v_{im}>\}$, où s_{ij}, $\forall\ j$ est un service déterministe virtuel qui représente le même service s_i. En outre, nous ajoutons un mapping entre chaque service virtuel s_{ij} et son service correspondant s_i

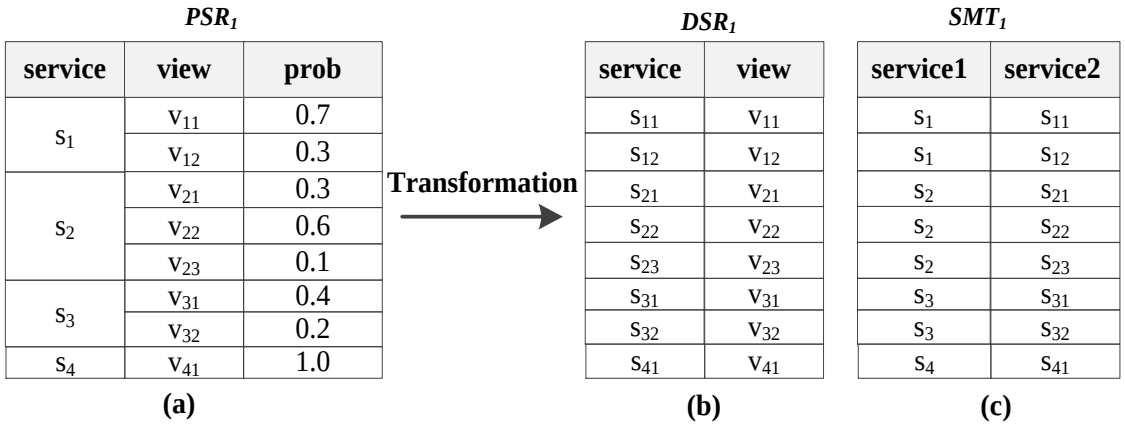


FIGURE 3.10 – (a) Registre de services probabiliste initial PSR_1. (b) Registre de services déterministes DSR_1. (c) Table de mapping SMT_1.

dans une table de mapping, appelée SMT. Notons que les probabilités de vues sémantiques sont ignorées dans cette phase, mais elles seront utilisées plus tard.

Example. Nous continuons avec notre exemple de motivation. Le registre de services probabiliste PSR_1 dans la figure 3.10(a) contient des services qui ont plusieurs vues sémantiques. PSR_1 est transformé en un registre de services déterministe DSR_1 représenté dans la figure 3.10(b). Chaque service dans DSR_1 a une seule vue sémantique. Par exemple, s_1 dans PSR_1 a deux vues sémantiques v_{11} et v_{12}, par conséquent, il est transformé en deux services différents : s_{11} associé à v_{11} et s_{12} associé à v_{12}. En outre, les relations entre les services de PSR_1 et ceux de DSR_1 sont stockées dans la table de mapping SMT_1 (voir la figure 3.10(c)). Par exemple, il existe des correspondances entre s_1 et s_{11}; s_1 et s_{12}; s_2 et s_{21}; s_2 et s_{22}, etc.

6.2 Réécriture de requêtes

En appliquant le processus de transformation sur le PSR, le nouveau registre DSR comprendra des services qui ont une seule vue sémantique. Cela nous permet de réécrire une requête Q à travers DSR, en utilisant un des algorithmes de réécriture proposés dans les environnements déterministes. Dans notre travail, nous utilisons l'algorithme de réécriture proposé dans [9]. Cet algorithme utilise les vues sémantiques des services pour réécrire une requête utilisateur comme étant un ensemble de compositions.

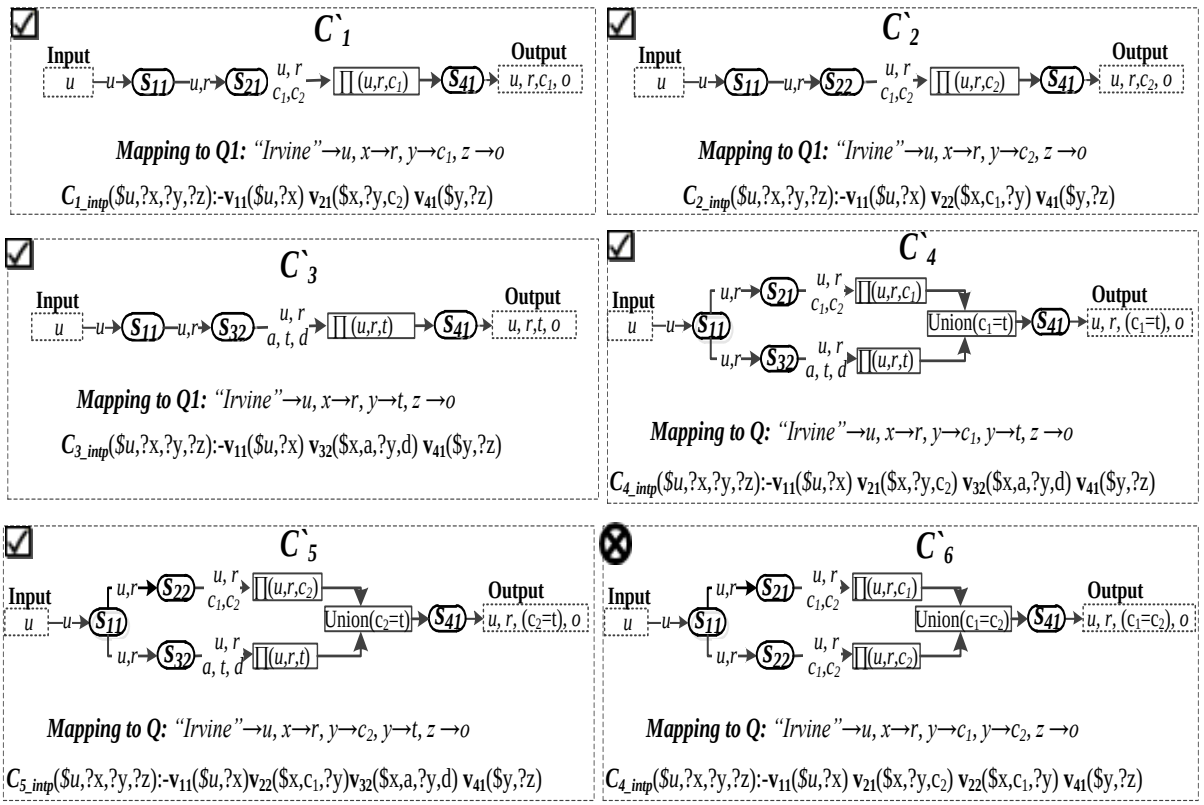


FIGURE 3.11 – Les compositions obtenues.

Le résultat de cette phase est un ensemble de compositions virtuelles [5] $C'=\{C'_1, C'_2, ..., C'_m\}$, tel que chaque composition virtuelle $C'_i \in C'$ a une interprétation C'_{i_intp} qui couvre Q.

Example. La réécriture de la requête Q_1 à travers DSR_1 (i.e., comme dans le cas certain) retourne les six compositions virtuelles illustrées dans la figure 3.11, chaque composition a une seule interprétation.

6.3 Elimination des fausses compositions

Après l'obtention de l'ensemble des compositions possibles, l'étape suivante consiste à éliminer les fausses compositions. Une composition est fausse si son interprétation contient des vues sémantiques appartenant au même bloc dans PSR (i.e., elles sont associées au même service dans PSR). Autrement dit, une composition est fausse si son interprétation

5. Nous appelons ces compositions *virtuelles* parce qu'elles impliquent des services virtuels

contient des *événements probabilistes disjoints*.

Par exemple, la composition C'_6 dans la figure 3.11 est éliminée, parce que son interprétation contient deux vues sémantique (v_{21}, v_{22}) qui appartiennent au même bloc dans PSR_1 (i.e., implique des événements probabilistes disjoints). Les autres composition (e.g., $C'_1, C'_2, C'_3, C'_4, C'_5$) contiennent uniquement des *événements probabilistes indépendants*, par conséquent elles sont correctes.

Définition 3.13 (***Fausse Composition***) : Soit $PSR = (pwd(SR), Pr)$ un registre de services probabiliste, et soit C'_k une composition possible associée à une interprétation C'_{k_intp}. La composition C'_k est fausse *ssi* $\exists < v_{ij}, v_{ih} > \in C'_{k\_intp}$, tel que $Pr(v_{ij}, v_{ih}) = 0$,i.e., v_{ij} et v_{ih} appartiennent au même bloc dans PSR, et par conséquent ils correspondent à des événements probabilistes disjoints.

6.4 Calcul des probabilités de compositions

Une fois l'ensemble des compositions obtenues est filtré en gardant uniquement celles qui sont correctes (i.e., correctes en termes d'événements probabilistes), la quatrième étape de notre méthode consiste à calculer la probabilité de chaque composition virtuelle C'_i. La probabilité d'une composition correspond à celle de son interprétation associée C'_{i_intp}. La probabilité d'une interprétation $Pr(C'_{i\_intp})$ peut être calculée efficacement (dans un temps polynomial) en utilisant le **Théorème 1** (i.e., en multipliant les probabilités des vues impliquées), et sans avoir besoin d'énumérer l'ensemble des registres de services possibles $pwd(SR)$.

La table 3.1 donne la probabilité de chaque interprétation de composition. Par exemple, l'interprétation C'_{2_intp} de la composition C'_2 implique trois vues sémantiques v_{11}, v_{22}, et v_{41}, ainsi sa probabilité est égale à $Pr(v_{11}) * Pr(v_{22}) * Pr(v_{41})$=0.7*0.6*1.0=0.42.

6.5 Transformation des compositions virtuelles

Les compositions obtenues ne peuvent pas être utilisées (ou, exécutées), parce que leurs plans contiennent des services virtuels (i.e., des services à partir de DSR). Pour cette raison, nous appliquons un autre processus de transformation sur les plans des compositions,

TABLE 3.1 – Probabilités des compositions

Interprétations	Probabilité
$C^{'}_{2\_intp}$	$Pr(v_{11}) * Pr(v_{22}) * Pr(v_{41}) = 0.42$
$C^{'}_{1\_intp}$	$Pr(v_{11}) * Pr(v_{21}) * Pr(v_{41}) = 0.21$
$C^{'}_{3\_intp}$	$Pr(v_{11}) * Pr(v_{32}) * Pr(v_{41}) = 0.14$
$C^{'}_{5\_intp}$	$Pr(v_{11}) * Pr(v_{22}) * Pr(v_{32}) * Pr(v_{41}) = 0.084$
$C^{'}_{4\_intp}$	$Pr(v_{11}) * Pr(v_{21}) * Pr(v_{32}) * Pr(v_{41}) = 0.042$

de sorte que, chaque composition virtuelle $C^{'}_{k}$ contenant des services virtuels est transformée en une composition exécutable C_k qui implique des services concrets. Autrement dit, chaque service virtuel s_{ij} dans une composition $C^{'}_{k}$ est remplacé par son correspondant s_i (i.e., le service initial) dans une composition exécutable C_k, tout en basant sur la table de mapping SMT. Ensuite, nous associons à la composition C_k une probabilité C_{k_pr} qui est égale à la probabilité de l'interprétation de $C^{'}_{k}$ (i.e. ; $Pr(C^{'}_{k\_intp})$). L'interprétation $C^{'}_{k\_intp}$ est l'une des interprétations possibles de C_k (i.e., $C^{'}_{k\_intp} \in pwd(C_{k\_intp})$), et C_k est une solution pour la requête Q, parce que $C^{'}_{k\_intp}$ couvre Q.

Example. En se basant sur la table de mapping SMT_1 dans la figure 3.10(c), la transformation des compositions virtuelles $\{C^{'}_1, C^{'}_2, C^{'}_3, C^{'}_4, C^{'}_5\}$ dans la figure 3.11 retourne le même ensemble de compositions que lorsque nous appliquons la *sémantique des compositions possibles* (figure 3.9(c)) (i.e., les mêmes plans de compositions, les mêmes probabilités). Cela montre que notre solution peut générer efficacement l'ensemble des compositions possibles, en accordance avec la *sémantique des compositions possibles*, sans avoir besoin de matérialiser l'ensemble des registres possibles.

Dans certains cas, la même composition possible peut être obtenue par la transformation de plus d'une composition virtuelle, i.e., deux (ou plusieurs) compositions virtuelles $C^{'}_{h}$ et $C^{'}_{z}$ peuvent correspondre à la même composition exécutable C_k. En d'autres termes, une composition C_k peut avoir plusieurs interprétations possibles couvrant la requête Q, et par conséquent, elles sont toutes pertinentes. Dans ce cas, la probabilité de C_k est égale à la somme des probabilités de toutes les compositions virtuelles qui lui correspondent.

Définition 3.14 (***Probabilité des Compositions finales***) : Soit $C^{'}$=$\{C^{'}_1, C^{'}_2, ..., C^{'}_m\}$ un ensemble de compositions virtuelles, et soit $Cs^{'} \subseteq C^{'}$, tel que $\forall\ C^{'}_h \in Cs^{'}$,

Possible Compositions	Obtained Data				
	University	Name	Cell Phone	Operator	Probability
C2	Irvine	Susan	040541268	Fr-Telecom	0.42
	Irvine	Andre	040541268	Fr-Telecom	0.42
	Irvine	Bob	040541268	Bouygtel	0.42
C1	Irvine	Susan	0797456589	Orange	0.21
	Irvine	Andre	0695487532	SFR	0.21
	Irvine	Bob	086214895	SFR	0.21

FIGURE 3.12 – Les données obtenues par C_1 et C_2

CompositionTransform(C'_h, SMT)=C_k [6], alors dans ce cas, C_{k_pr}=$\sum_{C'_h \in Cs'} Pr(C'_{h\_intp})$.

Une fois les compositions sont transformées, les utilisateurs peuvent choisir n'importe quelle composition possible pour l'exécuter. L'exécution d'une composition retourne un ensemble de tuples. Les données obtenues sont associés à la probabilité de la composition correspondante. L'utilisateur peut exécuter une seule composition à la fois (i.e., les compositions possibles sont exécutées séparément), comme il peut exécuter un ensemble de compositions, e.g., les $Top-k$ compositions classées en fonction de leurs probabilités. Par exemple, la figure 3.12 illustre les données obtenues par l'exécution du Top-2 compositions (i.e., C_2 et C_1) ; C_2 retourne pour le chercheur "Susan" un numéro de téléphone mobile "040541268" avec une probabilité 0,42 ; Tandis que C_1 retourne pour le même chercheur "Susan" un autre numéro "0797456589" avec une probabilité 0,21.

L'algorithme 1 décrit la réécriture des requêtes à travers un registre de services probabiliste, et de trouver efficacement l'ensemble des compositions possibles. Cet algorithme est divisé en cinq parties. La première partie (lignes 1-5) transforme le registre de services probabiliste PSR en un registre de services déterministe DSR, de sorte que chaque service s_i avec l'une de ses vues sémantiques v_{ij} (ligne 3) est transformé dans DSR comme un nouveau service s_{ij} qui a comme sémantique v_{ij} (ligne 4), de plus les correspondances entre s_i et ses nouveaux services $\{s_{i1}, s_{i2}, ..., s_{im}\}$ sont stockées dans la table de mapping SMT (ligne 5). Dans la deuxième partie (ligne 6), un algorithme de réécriture de requêtes QRA est utilisé pour générer les compositions candidates en reformulant Q via DSR. Dans la

6. CompositionTransform(C'_h, SMT) est une fonction qui remplace chaque service virtuel $s_{ij} \in C'_h$ par son correspondant $s_i \in C_k$ en se basant sur SMT

Algorithm 1: Génération des compositions possibles

Input: Q : Requête, PSR :Registre de services probabiliste.
Output: $Possible\_compo$: Ensemble des compositions possibles.

1 $DSR \longleftarrow \emptyset$; $SMT \longleftarrow \emptyset$;
2 **for** *each block* s_i *in* PSR **do**
3 **for** *each tuple* $< s_i, v_{ij} >$ *dans le block* s_i **do**
4 Adding $< s_{ij}, v_{ij} >$ to DSR ; // *Transformation*
5 Adding $< s_i, s_{ij} >$ to SMT ;
6 $Virtual\_compo \longleftarrow$ **QRA**(Q, DSR) ; // *Réécriture de requête*
7 **for** *each* $C_k^{'} \in Virtual\_compo$ **do**
8 **for** *each* $< v_{ij}, v_{xy} >$ *in* $C_{k\_intp}^{'}$ **do**
9 **if** *(*v_{ij} ***and*** v_{xy} *appartiennent au même bloc dans PSR)* **then**
10 $Virtual\_compo \longleftarrow Virtual\_compo - \{C_k^{'}\}$
11 **for** *each* $C_k^{'} \in Virtual\_compo$ **do**
12 $C_{k\_pr}^{'} \longleftarrow \prod_{v_{ij} \in C_{k\_intp}^{'}} Pr(v_{ij})$; // *Théorème 1*
13 **for** *each* $C_k^{'} \in Virtual\_compo$ **do**
14 $C_k \longleftarrow CompositionTransform(C_k^{'}, SMT)$;
15 **for** *each* $C_h^{'} \in Virtual\_compo$ **do**
16 **if** *(*$CompositionTransform(C_h^{'}, SMT) == C_k$*)* **then**
17 $C_{k\_pr} \longleftarrow C_{k\_pr} + C_{h\_pr}^{'}$;
18 $Virtual\_compo \longleftarrow Virtual\_compo - \{C_h^{'}\}$;
19 $Possible\_compo \longleftarrow Possible\_compo \cup \{< C_k, C_{k\_pr} >\}$;
20 return $Possible\_compo$;

troisième partie (lignes 7-10), les compositions virtuelles dont l'interprétation contient des événements probabilistes disjoints (ligne 9) sont ignorées (ligne 10). La quatrième partie (lignes 11-12) calcule pour chaque composition virtuelle correcte $C_k^{'}$ une probabilité $C_{k\_pr}^{'}$ en multipliant les probabilités des vues sémantiques impliquées (ligne 12). Dans l'étape finale (lignes 13-19), chaque composition virtuelle $C_k^{'}$ est transformée en une composition exécutable C_k (ligne 14). Si une composition C_k peut être obtenue par la transformation de plusieurs compositions virtuelles, alors la probabilité de C_K (i.e. ; C_{k_pr}) (lignes 15-18) est égale à la somme des probabilités de ces compositions virtuelles.

7 Optimisations

Dans cette partie, nous proposons quelques optimisations qui peuvent améliorer l'efficacité de l'Algorithme 1. Les optimisations proposées sont décrites dans les sous-sections

suivantes.

7.1 Elimination à priori des fausses compositions

Cette première optimisation concerne les fausses compositions qui sont actuellement éliminées après le calcul de toutes compositions candidates. Ces compositions peuvent être éliminées avant qu'elles ne soient produites. L'idée de base derrière cette optimisation consiste à étendre l'algorithme de réécriture [9] (i.e., la deuxième partie de l'Algorithme 1), de sorte qu'aucune fausse composition n'est produite.

L'algorithme de réécriture [9] est constitué principalement de deux phases : la *Phase de Sélection*, et la *Phase de Combinaison*. La phase de sélection retourne l'ensemble des services pertinents à la requête utilisateur Q, tel que chaque service pertinent s_{ik} couvre un ensemble de sous-buts B_{ik}=$\{b_1,...,b_t\}$ de Q (i.e., Les parties de Q qui sont couvertes par la vue sémantique v_{ik}). La phase de combinaison génère les compositions possibles en combinant les services pertinents (i.e., en se basant sur les sous-buts qu'ils couvrent), et en appliquant le *Test de Couverture de Requête* sur chaque combinaison (i.e., les combinaisons de services qui satisfont Q). L'idée est d'éliminer les fausses compositions avant que le *Test de Couverture de Requête* soit appliqué.

Une composition virtuelle C'_w est fausse si son interprétation contient des événements disjoints, i.e., si elle implique deux services virtuels s_{ik} et s_{ih} qui correspondent au même service initial s_i. Afin d'éliminer les fausses compositions avant l'application du *Test de Couverture de Requête*, nous étendons la *Phase de Combinaison* par l'ensemble de contraintes suivantes :

Soit s_{ik} et s_{ih} deux services virtuels qui correspondent au même service initial s_i. Soit C'_w une composition *encours de construction*, et qui implique s_{ik} et s_{ih} :

- Si $B_{ih} \subseteq B_{ik}$, alors supprimer s_{ih} de C'_w et continuer sa construction.
- Si $B_{ik} \cap B_{ih}$=$\emptyset$, alors arrêter la construction de C'_w et la supprimer.

L'impact de cette optimisation est montré dans le chapitre d'évaluation.

7.2 Génération efficace des Top-k compositions

Une des approches qui peut être utilisée pour calculer les *Top-k* compositions, est de générer toutes les compositions possibles, calculer leurs probabilités, et enfin retourner

les k meilleures compositions en se basant sur leurs probabilités. Cependant, il est clair que cette approche est naïve, et coûteuse en temps d'exécution, puisque elle doit générer toutes les compositions possibles, alors que la plupart d'entre elles ne font pas partie des k compositions les plus probables.

Dans ce qui suit, nous proposons une technique d'optimisation qui permet de calculer efficacement les k meilleures compositions. L'idée est d'étendre l'algorithme de réécriture de requêtes (i.e., la deuxième partie de l'Algorithme 1) de façon à supprimer les services (virtuels) qui s'ils sont composés avec d'autres services, les compositions obtenues n'appartiendraient pas à l'ensemble des *Top-k* compositions. Plus spécifiquement, les services pertinents (virtuels) qui sont retournés par la première phase (phase de sélection) de l'algorithme de réécriture sont regroupés en un ensemble de classes SC=$\{SC_1, SC_2,\ldots,SC_n\}$, tel que les services dans chaque classe SC_h couvrent la même partie (sous-requête) q_h de la requête Q. Ensuite, les services s_{ij} dans chaque classe SC_h sont classés en se basant sur les probabilités de leurs vues sémantiques, et seuls les *Top-k* services dans chaque classe sont retenus. Les services retenus sont ensuite utilisés pour calculer les compositions possibles ainsi que leurs probabilités, et les *Top-k* compositions peuvent être retournées aux utilisateurs.

L'Algorithme 2 permet de générer efficacement l'ensemble des *Top-k* compositions. Dans ce qui suit, nous décrivons les principales étapes de notre algorithme.

– **Etape 1** *Construction des classes (lignes 1-9).* Dans cette phase, les services virtuels qui couvrent la requête utilisateur de la même façon (i.e., couvrent les mêmes sous-buts ligne 6) sont regroupés dans une même classe SC_h (ligne 7).

– **Etape 2** *Génération des Top-k services dans chaque classe (lignes 10-11).* Dans chaque classe de services SC_h, les services sont classés en se basant sur la probabilité des vues sémantiques associées. Ensuite, pour chaque classe SC_h nous créons une nouvelle classe $Top\text{-}k.SC_h$ (ligne 11) qui contient que les meilleurs k services.

– **Etape 3** *Génération des Top-k compositions (lignes 12-17).* Nous générons les compositions virtuelles qui couvrent la requête utilisateur Q (ligne 12), tout en se basant uniquement sur les Top-k services dans chaque classe. Pour chaque composition virtuelle, nous calculons sa probabilité (ligne 15), et ensuite nous appliquons sur chacune un processus de transformation (ligne 16) (i.e., comme dans l'Algorithme 1). Enfin, nous

Algorithm 2: Génération des Top-k compositions

Input: Q : Requête, S=$\{<s_{ij}, B_{ij}>,...,<s_{mn}, B_{mn}>\}$:ensemble de services (virtuels) pertinents., $k \in \mathbb{N}$

Output: Ensemble des Top-k compositions.

```
1  SC ⟵ ∅ ;
2  for each service s_ij in S do
3      SC_h ⟵ {s_ij} ;
4      S ⟵ S - {s_ij} ;
5      for each service s_kz in S do
6          if (s_kz.B_kz == s_ij.B_ij) then
7              SC_h ⟵ SC_h ∪ {s_kz} ;
8              S ⟵ S - {s_kz} ;
9      SC ⟵ SC ∪ {SC_h} ;
10 for each service class SC_h in SC do
11     Top-k.SC_h ⟵ top(k,SC_h) ;
12 Possible_Virtual_Composition ⟵ CombinationProcess(Q,{Top-k.SC_1,...,Top-k.SC_n} )
13 Possible_Composition ⟵ ∅ ;
14 for each C'_k ∈ Possible_Virtual_Composition do
15     ComputingCompositionProbabilities(C'_k) ; //en utilisant Théorème 1
16     Possible_Composition ⟵ Possible_Composition ∪ CompositionTransformation(C'_k) ;
17 return top(k,Possible_Composition) ;
```

ne retournons aux utilisateurs que les K meilleures compositions en se basant sur leurs probabilités.

Exemple. Nous continuons avec notre exemple, Les services (virtuel) pertinents sont regroupés dans trois classes SC_1 =$\{s_{11}\}$, SC_2 =$\{s_{22}, s_{21}, s_{32}\}$, et SC_3 =$\{s_{41}\}$ qui correspondent respectivement aux sous-requêtes suivantes (exprimées en Datalog)

q_1 :- $U(\text{“}Irvive\text{”}), current(U,T)$, $T(x)$

q_2 :- $T(x,y)$

q_3 :- $O(z), subscriber(O,C)$, $C(y)$

Si nous sommes intéressés pour calcul des Top-1 compositions, alors les top-1 services des trois classes sont s_{11}, s_{22} et s_{41}. La combinaison de ces services retourne la composition C_2, avec une probabilité égale à 0.42.

8 Conclusion

Dans ce chapitre, nous avons proposé une approche probabiliste qui permet la représentation et la composition des services DaaS à sémantique incertaine. En particulier, nous avons

proposé le concept Registre de Services Probabiliste qui permet de représenter un ensemble de services avec leurs vues sémantiques possibles. Ensuite, nous avons défini la façon dont les différentes interprétations possibles d'une composition existante (impliquant des services DaaS incertains) peuvent être calculées. Nous avons également défini un algorithme qui assure l'évaluation efficace des requêtes utilisateurs.

Le registre de services probabiliste PSR est formellement décrit par le modèle BID, de sorte que les vues sémantiques des services différents sont considérées comme des événements probabilistes indépendants (pas de corrélation entre elles). Cependant, dans la pratique, la même source de données peut être encapsulée par plusieurs services, et par conséquent, les vues sémantiques possibles de ces derniers ne sont pas complétement indépendantes (corrélées). Dans le chapitre suivant, nous proposons une approche plus générique qui permet de représenter les différentes corrélations qui peuvent exister entre les vues sémantiques possibles. Nous proposons également une méthode efficace pour l'évaluation des requêtes à travers des services à sémantique corrélée.

Chapitre 4

Modélisation et composition des services à sémantique corrélée

1 Introduction

Dans le chapitre précédent, nous avons proposé une approche pour la modélisation et la composition des services DaaS à sémantique incertaine. Chaque service s_i dont la sémantique est incertaine est associé à un ensemble de vues sémantiques possibles, chacune avec une probabilité. Les services ainsi que leurs vues sémantiques possibles sont décrits dans un registre de services probabiliste PSR, dans lequel un service avec une de ses vues sémantiques possibles est considéré comme étant un tuple. Chaque vue sémantique v_{ij} d'un service s_i (ou un tuple dans PSR) est considérée comme étant un *événement probabiliste*, tel que v_{ij} a une probabilité $Pr(v_{ij})$. Le registre de services probabiliste PSR est formellement décrit par le modèle BID, tel que les vues sémantiques du même service sont considérées comme des événements probabilistes disjoints (i.e., une corrélation de type *exclusion mutuelle*), alors que les vues sémantiques des services différents sont considérées comme des événements probabilistes indépendants (pas de corrélation entre eux). Dans le modèle BID, il est supposé que les sources de données encapsulées par l'ensemble des services sont indépendantes, ce qui explique la non-corrélation entres les vues sémantiques des services différents.

Cependant, dans la pratique, la même source de données peut être encapsulée par plu-

Services DaaS	Fonctionnalités
s_1($u, ?n)	Retourne les noms des chercheurs (?n) pour une université donnée ($u).
s_2($n, ?c1, ?c2)	Retourne les contactes (?c1, ?c2) d'un chercheur ($n).
s_3($n, ?c1, ?e)	Retourne les contactes (?c1, ?e) d'un chercheur ($n).

(a)

T1

u	n
Irvine	Susan
Irvine	Andre
Irvine	Bob
....	....

T2

n	c1	c2	e
Susan	0797456789	040542189	Susan@gmail.com
Andre	0696125831	042569874	Andre@Irvine.edu
Bob	0745369825	021459854	Bob@yahoo.com
....	....	....	....

(b)

(c)

Vues sémantiques possibles de s_1($u, ?n)		Description	Prob
v_{11}	Select ?n Where{?U :rdftype :University ?U :hasCode $u, ?U ***:current*** ?T ?T :rdftype :Researcher, ?T :HasName ?n	Retourne les noms (?n) des chercheurs *actuels* à l'université ($u)	0.7
v_{12}	Select ?n Where{?U :rdftype :University ?U :HasCode $u, ?U ***:former*** ?T ?T :rdftype :Researcher, ?T :HasName ?n	Retourne les noms (?n) des *anciens* chercheurs de l'université ($u)	0.3

Vues sémantiques possibles de s_2($n, ?c1, ?c2)		Description	Prob
v_{21}	Select ?c1, ?c2 Where{?T :rdftype :Researcher, ?T :hasName $n ?T *:cellPhone* ? c1 ?T *:homePhone* ?c2	Retourne le *téléphone mobile* (?c1), et *domicile* (?c2), d'un chercheur ($n)	0.5
v_{22}	Select ?c1, ?c2 Where{?T :rdftype :Researcher, ?T :hasName $n ?T *:homePhone* ?c1 ?T *:cellPhone* ?c2	Retourne le *téléphone domicile* (?c1), et *mobile* (?c2), d'un chercheur ($n)	0.2

Vues sémantiques possibles de s_3($n, ?c1, ?e)		Description	Prob
v_{31}	Select ?c1, ?e Where{?T :rdftype :Researcher, ?T :hasName $n ?T *:cellPhone* ? c1 ?T :hasEmail ?e	Retourne le *téléphone mobile* (?c1), et l'Email (?e) d'un chercheur ($n)	0.4
v_{32}	Select ?c1, ?e Where{?T :rdftype :Researcher, ?T :hasName $n ?T *:fax* ? c1, ?T :hasEmail ?e	Retourne le *fax* (?c1), et l'Email (?e) d'un chercheur ($n)	0.2

FIGURE 4.1 – (a) Services DaaS. (b)Les données encapsulées par les services. (c)Les vues sémantiques possibles de chaque service.

sieurs services (e.g., deux services qui partagent partiellement ou complétement les mêmes sources de données), et par conséquent, les vues sémantiques possibles de ces derniers ne sont pas complétement indépendantes. Dans ce cas, différentes corrélations (généralement complexes) peuvent exister entre les vues sémantiques des services différents. Une corrélation entre deux vues sémantiques permet de définir toutes les dépendances probabilistes qui existent entre les événements associés. Ces dépendances probabilistes sont généralement décrites par des probabilités conditionnelles qui permettent à une source de données d'être interprétée de façon cohérente et non-ambiguë. Ceci est particulièrement fréquent lorsque deux ou plusieurs services qui partagent la même source de données sont impliqués dans la même composition. Dans une telle situation, les vues sémantiques associées aux services impliqués (l'interprétation de la composition) doivent interpréter de façon cohérente les données communes.

1.1 Exemple de motivation

Considérons les services DaaS de la figure 4.1 (a). Ces services sont construits en utilisant les sources de données de la figure 4.1 (b). Supposons qu'un processus de sémantisation (à travers une ontologie de domaine, voir la Figure 3.2) a été réalisé sur ces services en utilisant un système de matching semi-automatique. Dans cet exemple, les interprétations de ces services sont incertaines, ce qui permet d'associer à chacun un ensemble de vues sémantiques probabilistes (voir la figure 4.1 (C)). Par exemple, le service s_2 est associé à deux vues v_{21} et v_{22} dont les probabilités sont respectivement 0.5 et 0.2.

Une première corrélation intuitive qui peut être définie entre les vues sémantiques est l'*exclusion mutuelle* : les vues sémantiques possibles d'un même service sont mutuellement exclusives, alors que les vues sémantiques de services différents sont supposées indépendantes, i.e., il n'y a pas de corrélation entre elles. Par exemple, les vues sémantiques v_{11} et v_{12} sont mutuellement exclusives, puisque elles sont associées au même service s_1. Les vues sémantiques v_{11} et v_{21} sont indépendantes, parce qu'elles sont associées à des services différents. Cependant, certains services peuvent partager les mêmes sources de données, et par conséquent leurs vues sémantiques ne sont pas totalement indépendantes (i.e., elles sont corrélées). Par exemple, les services s_2 et s_3 encapsulent la même source de données $T2(n, c1, c2, e)$ (figure 4.1 (b)), ils partagent entre eux les colonnes $T2.n$ et $T2.c1$. Dans ce cas, si s_2 et s_3 sont impliqués dans la même composition, les vues sémantiques associées [1] doivent interpréter $T2.n$ et $T2.c1$ de la même manière, i.e., s_2 et s_3 ne peuvent pas être associés respectivement, avec v_{21} et v_{32}, parce qu'elles interprètent la même entité $T2.c1$ de façon contradictoire (i.e., *téléphone mobile* dans v_{21}, et *fax* dans v_{32}), par conséquent v_{21} et v_{32} ne sont pas indépendantes, et il est nécessaire de préciser leur corrélation.

1.2 Challenges

L'exemple de motivation ci-dessus pose les challenges suivants :

– *Comment représenter les corrélations qui existent entre les sémantiques des services* : Lorsque les services DaaS encapsulent les mêmes sources de données, il est nécessaire de maintenir les dépendances probabilistes qui existent entre leurs vues sémantiques

1. Dans une composition, un service est associé à une de ses vues sémantiques possibles

possibles. Cependant, comment modéliser ces dépendances de façon que nous puissions représenter (de manière générique) n'importe quelles corrélations complexes.

– *Comment évaluer une requête à travers des services à sémantique corrélée* : Dans le chapitre 3, nous avons proposé un algorithme qui permet d'évaluer efficacement une requête utilisateur. Cet algorithme calcule la probabilité des compositions obtenues en supposant qu'il y a un seul type de corrélation entre l'ensemble des vues sémantiques (i.e., l'exclusion mutuelle). Cependant, lorsque les vues sémantiques sont corrélées de différentes manières, l'algorithme de réécriture doit être étendu de façon à préserver (respecter) ces corrélations pendant le processus de réécriture (i.e., en évitant les compositions qui ne satisfont pas les corrélations).

1.3 Contributions

Dans ce chapitre, nous proposons une approche probabiliste (à base de réseaux bayésiens) pour la représentation des services dont les sémantiques sont corrélées [75]. Ci-dessous, nous résumons nos principales contributions :

– *Représenter PSR à travers un réseau bayésien* : Les corrélations dans le registre de services probabiliste PSR peuvent être représentées de façon générique en utilisant les réseaux bayésiens (i.e., graphe probabiliste orienté). Cette représentation est réalisée en introduisant des variables aléatoires, et des distributions de probabilités conditionnelles entre elles.

– *Evaluation de requêtes* : Evaluer une requête à travers un PSR corrélé (représenté par un réseau bayésien), revient à l'évaluer sur chaque registre de services possible, comme dans le modèle BID. Pour l'évaluation efficace des requêtes, nous appliquons presque les mêmes étapes de l'Algorithme 1 du chapitre précédent. Néamoins, le **Théorème 1** n'est plus applicable pour le calcul des probabilités des compositions obtenues. La probabilité de chaque composition est calculée en se basant sur un algorithme d'inférence probabiliste, appelé *Variables Elimination* (VE) [28]. Cet algorithme infère à partir d'un réseau bayésien la probabilité marginale d'une interprétation de composition, en considérant cette dernière comme étant un sous-ensemble de variables aléatoires.

Le reste de ce chapitre est organisé comme suit. Dans la section 2, nous décrivons la représentation bayésienne d'un registre de services probabiliste. Dans la section 3, nous

définissons la sémantique de l'évaluation d'une requête à travers un PSR corrélé. La section 4 présente une approche qui permet l'évaluation efficace des requêtes, tout en introduisant un algorithme d'inférence probabiliste. Enfin, dans la section 5, nous concluons ce chapitre.

2 Représentation des corrélations dans PSR

Comme nous l'avons expliqué dans l'exemple de motivation, la corrélation entre les services se produit naturellement dans de nombreux domaines d'application, et ignorer une telle corrélation peut entraîner des résultats imprécis et ambigus. Plusieurs modèles ont été proposés pour représenter la corrélation qui existe entre les tuples d'une base de données probabiliste [5, 24, 108]. Parmi ces modèles, les réseaux bayésiens sont les plus convenables pour représenter les différentes corrélations complexes qui existent entre les vues sémantiques dans un registre PSR.

Les corrélations dans PSR peuvent être modélisées de manière générique en utilisant les réseaux bayésiens, de telle sorte que chaque tuple (ou une vue sémantique) dans PSR est associé à une variable aléatoire booléenne qui décrit l'incertitude de son existence. Dans PSR, la présence d'un tuple t_{ij} signifie que le service s_i est sémantiquement décrit par la vue sémantique v_{ij}. En revanche, l'absence de t_{ij} signifie que le service s_i est sémantiquement décrit par une vue sémantique différente de v_{ij}. Représenter la corrélation entre deux vues sémantiques revient à définir une distribution de probabilité conditionnelle entre les variables aléatoires associées.

Définition 4.1 (***Représentation bayésienne de PSR***) : Un registre de services probabiliste PSR est formellement représenté par un réseau bayésien BN :(X, E, $\mathcal{P}$) tel que :

– Chaque tuple t_{ij} :<s_i, v_{ij}, $Pr(v_{ij})$> $\in$ PSR est associé à une variable aléatoire booléenne $x_{ij} \in X$ (i.e., $dom(x_{ij})$={0,1}), tel que, si v_{ij} décrit la sémantique de s_i (i.e., t_{ij} est présent), alors x_{ij}=1, sinon x_{ij}=0 (i.e., t_{ij} est absent).

– S'il existe des corrélations orientées entre les vues sémantiques {v_{kh},...,v_{mn}} et v_{ij}, alors : (1) ajouter dans E les arcs {(x_{kh}, x_{ij}),...,(x_{mn}, x_{ij})} reliant les variables qui correspondent aux tuples de {v_{kh},...,v_{mn}} et celui de v_{ij} ; (2) Associer au nœud de x_{ij} une distribution de probabilité conditionnelle $Pr(x_{ij}|par(x_{ij})) \in \mathcal{P}$ (i.e., $par(x_{ij})$={x_{kh},...,x_{mn}})

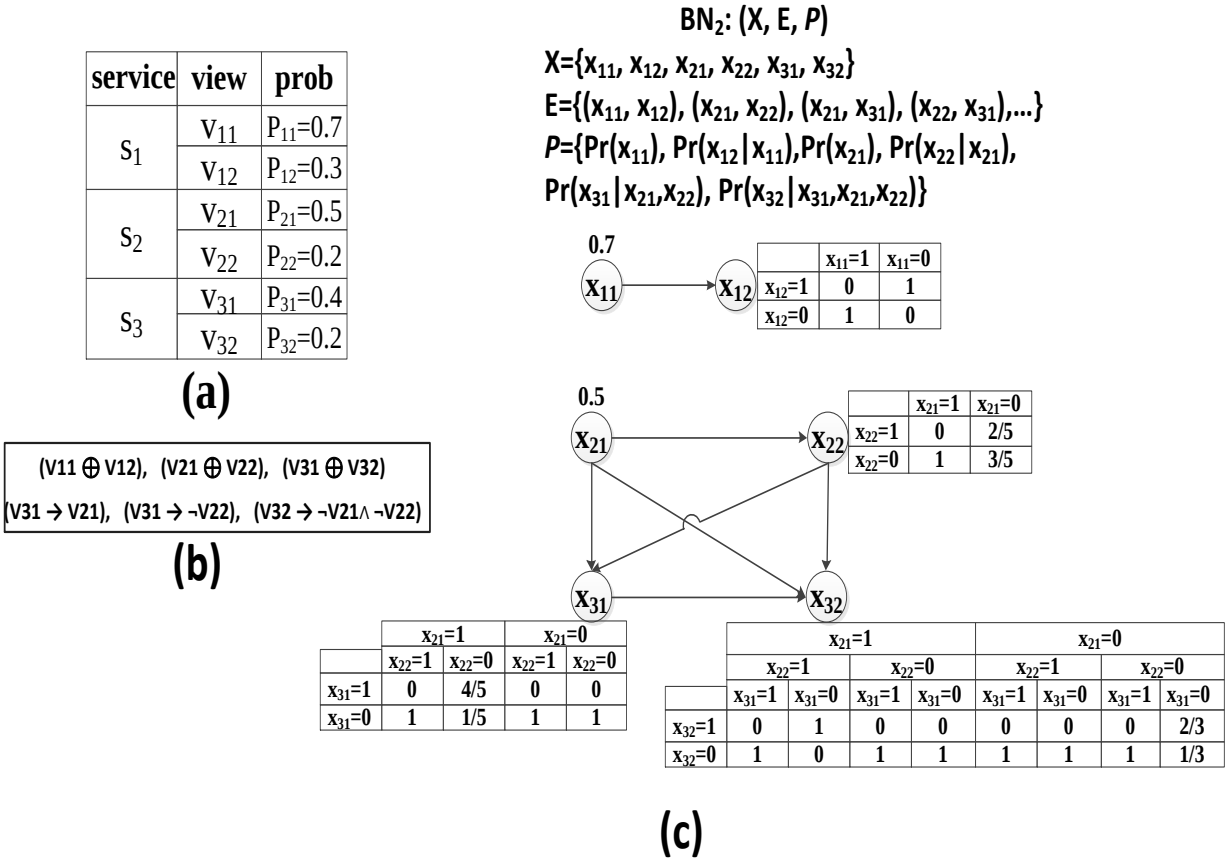


FIGURE 4.2 – (a) Registre de services probabiliste PSR_2.(b)Les corrélation dans PSR_2, (c)PSR_2 avec la représentation bayésienne

quantifiant les corrélations qui existent entre les vues sémantiques $\{v_{kh},...,v_{mn}\}$ et v_{ij}.

– S'il existe un nœud x_{ij} qui n'a pas de prédécesseurs dans le graphe de BN (i.e., $par(x_{ij})=\emptyset$), alors il est associé às la probabilité marginale $Pr(v_{ij})$ de la vue sémantique correspondante.

Example. La figure 4.2(a) montre le registre de services probabiliste PSR_2 qui correspond aux services incertains de notre exemple de motivation. Plusieurs corrélations peuvent exister entre les vues sémantiques dans PSR_2 (voir la fiigure 4.2(b)). Par exemple : $v_{11} \oplus v_{12}$ (i.e., mutuellement exclusives), parce qu'elles sont associées au même service s_1 ; $v_{31} \rightarrow v_{21}$ (i.e., v_{31} implique v_{21}), puisque les données retournées par s_3 sont incluses dans celles retournées par s_2, de plus v_{31} et v_{21} ne sont pas sémantiquement contradictoires (i.e., $?c1$ comme $cellPhone$) ; etc.

Registres possibles	Assignements complets	Probabilités jointes
SR1= **service** | **view** S_1 | V_{11} S_2 | V_{21} S_3 | V_{31}	$\mathbf{AX_1}$={x_{11}=1, x_{12}=0, x_{21}=1, x_{22}=0, x_{31}=1, x_{32}=0}	Pr(x_{11}=1)*Pr(x_{12}=0|x_{11}=1)*Pr(x_{21}=1)*Pr(x_{22}=0|x_{21}=1)* Pr(x_{31}=1| x_{21}=1, x_{22}=0)*Pr(x_{32}=0|x_{31}=1,x_{21}=1,x_{22}=0) = 0.7* 1* 0.5* 1* 4/5*1=0.28.
SR2= **service** | **view** S_1 | V_{11} S_2 | V_{21}	$\mathbf{AX_2}$={x_{11}=1, x_{12}=0, x_{21}=1, x_{22}=0, x_{31}=0, x_{32}=0}	Pr(x_{11}=1)*Pr(x_{12}=0|x_{11}=1)*Pr(x_{21}=1)*Pr(x_{22}=0|x_{21}=1)* Pr(x_{31}=0| x_{21}=1, x_{22}=0)*Pr(x_{32}=0|x_{31}=0,x_{21}=1,x_{22}=0) = 0.7* 1* 0.5* 1* 1/5*1=0.07.
SR3= **service** | **view** S_1 | V_{11} S_2 | V_{22}	$\mathbf{AX_3}$={x_{11}=1, x_{12}=0, x_{21}=0, x_{22}=1, x_{31}=0, x_{32}=0}	Pr(x_{11}=1)*Pr(x_{12}=0|x_{11}=1)*Pr(x_{21}=0)*Pr(x_{22}=1|x_{21}=0)* Pr(x_{31}=0| x_{21}=0, x_{22}=1)*Pr(x_{32}=0|x_{31}=0,x_{21}=0,x_{22}=1) = 0.7* 1* 0.5* 2/5* 1*1=0.14.
SR4= **service** | **view** S_1 | V_{11} S_3 | V_{32}	$\mathbf{AX_4}$={x_{11}=1, x_{12}=0, x_{21}=0, x_{22}=0, x_{31}=0, x_{32}=1}	Pr(x_{11}=1)*Pr(x_{12}=0|x_{11}=1)*Pr(x_{21}=0)*Pr(x_{22}=0|x_{21}=0)* Pr(x_{31}=0| x_{21}=0, x_{22}=0)*Pr(x_{32}=1|x_{31}=0,x_{21}=0,x_{22}=0) = 0.7* 1* 0.5* 3/5* 1*2/3=0.14.
SR5= **service** | **view** S_1 | V_{11}	$\mathbf{AX_5}$={x_{11}=1, x_{12}=0, x_{21}=0, x_{22}=0, x_{31}=0, x_{32}=0}	Pr(x_{11}=1)*Pr(x_{12}=0|x_{11}=1)*Pr(x_{21}=0)*Pr(x_{22}=0|x_{21}=0)* Pr(x_{31}=0| x_{21}=0, x_{22}=0)*Pr(x_{32}=0|x_{31}=0,x_{21}=0,x_{22}=0) = 0.7* 1* 0.5* 3/5* 1*1/3=0.07.
SR6= **service** | **view** S_1 | V_{12} S_2 | V_{21} S_3 | V_{31}	$\mathbf{AX_6}$={x_{11}=0, x_{12}=1, x_{21}=1, x_{22}=0, x_{31}=1, x_{32}=0}	Pr(x_{11}=0)*Pr(x_{12}=1|x_{11}=0)*Pr(x_{21}=1)*Pr(x_{22}=0|x_{21}=1)* Pr(x_{31}=1| x_{21}=1, x_{22}=0)*Pr(x_{32}=0|x_{31}=1,x_{21}=1,x_{22}=0) = 0.3* 1* 0.5* 1* 4/5*1=0.12.
SR7= **service** | **view** S_1 | V_{12} S_2 | V_{21}	$\mathbf{AX_7}$={x_{11}=0, x_{12}=1, x_{21}=1, x_{22}=0, x_{31}=0, x_{32}=0}	Pr(x_{11}=0)*Pr(x_{12}=1|x_{11}=0)*Pr(x_{21}=1)*Pr(x_{22}=0|x_{21}=1)* Pr(x_{31}=0| x_{21}=1, x_{22}=0)*Pr(x_{32}=0|x_{31}=0,x_{21}=1,x_{22}=0) = 0.3* 1* 0.5* 1* 1/5*1=0.03.
SR8= **service** | **view** S_1 | V_{12} S_2 | V_{22}	$\mathbf{AX_8}$={x_{11}=0, x_{12}=1, x_{21}=0, x_{22}=1, x_{31}=0, x_{32}=0}	Pr(x_{11}=0)*Pr(x_{12}=1|x_{11}=0)*Pr(x_{21}=0)*Pr(x_{22}=1|x_{21}=0)* Pr(x_{31}=0| x_{21}=0, x_{22}=1)*Pr(x_{32}=0|x_{31}=0,x_{21}=0,x_{22}=1) = 0.3* 1* 0.5* 2/5* 1*1=0.06.
SR9= **service** | **view** S_1 | V_{12} S_3 | V_{32}	$\mathbf{AX_9}$={x_{11}=0, x_{12}=1, x_{21}=0, x_{22}=0, x_{31}=0, x_{32}=1}	Pr(x_{11}=0)*Pr(x_{12}=1|x_{11}=0)*Pr(x_{21}=0)*Pr(x_{22}=0|x_{21}=0)* Pr(x_{31}=0| x_{21}=0, x_{22}=0)*Pr(x_{32}=1|x_{31}=0,x_{21}=0,x_{22}=0) = 0.3* 1* 0.5* 3/5* 1*2/3=0.06.
SR10= **service** | **view** S_1 | V_{12}	$\mathbf{AX_{10}}$={x_{11}=0, x_{12}=1,x_{21}=0, x_{22}=0, x_{31}=0, x_{32}=0}	Pr(x_{11}=0)*Pr(x_{12}=1|x_{11}=0)*Pr(x_{21}=0)*Pr(x_{22}=0|x_{21}=0)* Pr(x_{31}=0| x_{21}=0, x_{22}=0)*Pr(x_{32}=0|x_{31}=0,x_{21}=0,x_{22}=0) = 0.3* 1* 0.5* 3/5* 1*1/3=0.03.

FIGURE 4.3 – Les registres de services possibles de PSR_2.

PSR_2 avec ces corrélations sont représentées à travers le réseau bayésien BN_2 (voir la figure 4.2(c)), dans lequel chaque tuple dans PSR_2 est représenté par une variable aléatoire booléenne, e.g., $<s_1, v_{11}, 0.7>$ comme x_{11}, $<s_2, v_{22}, 0.2>$ comme x_{22}, etc.

Les corrélations sont maintenues en ajoutant des arcs dans le graphe de BN_2 (e.g., (x_{11}, x_{12}) puisque $v_{11} \oplus v_{12}$), et en associant à chaque nœud une table qui décrit la distribution de probabilité conditionnelle sachant ses prédécesseurs. Par exemple, v_{21} et v_{22} sont corrélées avec v_{31}, tel que, les prédécesseurs de x_{31} sont x_{21} et x_{22}. Dans ce cas, la variable x_{31} est associée à une table conditionnelle représentant la distribution de probabilité conditionnelle $Pr(x_{31}|x_{21}, x_{22})$. Les nœuds qui n'ont pas de prédécesseurs sont associés à leurs probabilités marginales (e.g., x_{11} avec 0.7, x_{21} avec 0.5).

Comme dans le modèle BID, la sémantique de PSR peut être définie en se basant sur la *Sémantique des mondes possibles* [25], où PSR est considéré comme étant une distribution de probabilité sur un ensemble de registres de services déterministes $pwd(SR)$=$\{SR_1, ..., SR_n\}$. $pwd(SR)$ est construit en générant les différentes combinaisons possibles des tuples dans PSR. Chaque registre de services possible SR_k peut être exprimé par un assignement complet AX_k de l'ensemble des variables aléatoires X, i.e., $AX_k \in \{0,1\}^{|X|}$. Calculer la probabilité de SR_k revient à calculer la probabilité jointe de l'assignement complet AX_k. Cette probabilité jointe est calculée en multipliant les valeurs retournées par toutes les distributions conditionnelles (sachant AX_k) dans $\mathcal{P}$.

Définition 4.2 (***Sémantique de PSR***) : Soit PSR=($pwd(SR)$, BN) un registre de services probabiliste, où $pwd(SR)$=$\{SR_1, SR_2, ..., SR_n\}$ est l'ensemble des registres de services possibles de PSR, et BN :(X, E, $\mathcal{P}$) est un réseau bayésien qui décrit les corrélations qui existent dans PSR. BN définit une distribution de probabilité sur $pwd(SR)$. Chaque registre de services possible $SR_k \in pwd(SR)$ peut être exprimé par un assignement complet $AX_k \in \{0,1\}^{|X|}$, tel que, si un tuple $<s_i, v_{ij}, Pr(v_{ij})> \in SR_k$, alors la variable aléatoire correspondante x_{ij} est assignée à 1 dans AX_k, sinon elle assignée à 0. Par conséquent, la probabilité de SR_k est égale à la probabilité jointe de AX_k, et est calculée comme suit :

- $Pr(SR_k)$=$Pr_{BN}(X \sim AX_k)$=$\prod_{x_i \in X} Pr(x_i|\ par(x_i))$. / $Pr(x_i|\ par(x_i)) \in \mathcal{P}$
- $\sum_{SR_k \in pwd(SR)} Pr(SR_k) = 1$.

Example. La figure 4.3 montre les dix registres de services possibles qui correspondent à

PSR_2 dans la figure 4.2(a). SR_1 implique les trois services comme suit : s_1 avec v_{11}, s_2 avec v_{21}, et s_3 avec v_{31}. En se basant sur le réseau bayésien BN_2 de la figure 4.2(c), SR_1 est associé avec l'assignement complet AX_1 :{x_{11}=1, x_{12}=0, x_{21}=1, x_{22}=0, x_{31}=1, x_{32}=0}, et dont la probabilité jointe est calculée comme suit :

$Pr(SR_1)$= $Pr(x_{11}$=1)*$Pr(x_{12}$=0|x_{11}=1)*$Pr(x_{21}$=1)*$Pr(x_{22}$=0|x_{21}=1)* $Pr(x_{31}$=1| x_{21}=1, x_{22}=0)*
$Pr(x_{32}$=0|x_{31}=1,x_{21}=1,x_{22}=0)=0.7* 1* 0.5* 1* $\frac{4}{5}$*1=0.28.

3 Evaluation de requêtes à travers un PSR corrélé

Dans le modèle BID, évaluer (ou réécrire) une requête Q à travers un registre de services probabiliste PSR revient à l'évaluer sur tous les registres de services possibles de PSR. Ensuite, les résultats obtenus peuvent être interprétés de deux façons différentes : soit par la *Sémantique des Résultats possibles*, soit par la *Sémantique des Compositions possibles*.

Dans le cas où PSR est décrit par un réseau bayésien, ce qui change par rapport au modèle BID, c'est uniquement les registres de services possibles (leurs contenus, et leurs probabilités). Par conséquent, l'évaluation d'une requête Q à travers un PSR bayésien est réalisée de la même façon que lorsque PSR est décrit par le modèle BID. Cependant, les résultats obtenus sont souvent différents, tout simplement parce que, la requête Q n'est pas évaluée sur le même ensemble de registres possibles (contenus différents, probabilités différentes, etc).

Comme dans le chapitre précédent, dans cette partie, nous adaptons la *Sémantique des Compositions possibles* pour interpréter les résultats de l'évaluation des requêtes. Dans ce cas, la réécriture d'une requête Q à travers un registre de services probabiliste PSR=($pwd(SR)$, BN) permet de retourner un ensemble de compositions possibles $Q^{rank}(PSR) = \{C_1, C_2, ..., C_w\}$. Chaque composition $C_i \in Q^{rank}(PSR)$ a une interprétation $C_{i\_intp\_x}$ qui couvre Q, et dont la probabilité est égale à la somme des probabilités des registres de services possibles qui retournent C_i (ou les résultats possibles qui impliquent C_i).

Exemple. Supposons qu'un utilisateur veuille trouver les numéros de téléphone mobiles des chercheurs actuels de l'université Irvine. Pour cette raison, il soumet la requête

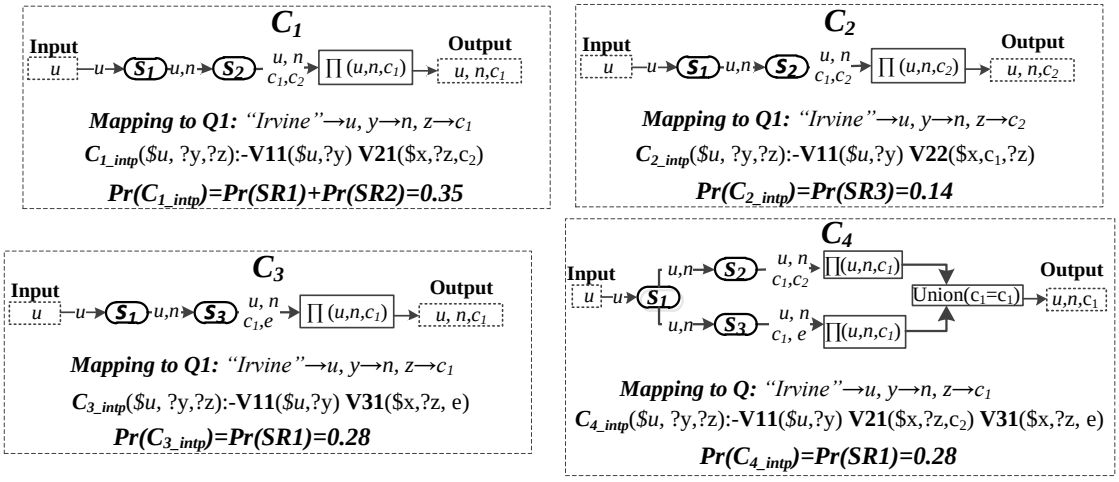


FIGURE 4.4 – Les compositions possibles de Q_2.

SPARQL suivante (chaque triplet RDF est considéré comme un prédicat) : `Q2(?y,?z): University(?U) hasCode("Irvine") current(?U,?T) Researcher(?T) hasName(?T,?y) CellPhone(?T,?z)`

En se basant sur la *Sémantique des Compositions possibles*, réécrire Q_2 à travers PSR_2 permet de retourner quatre compositions$Q^{rank}(PSR)$=$\{C_1, C_2, C_3, C_4\}$ (voir la figure 4.4). La composition C_1 appartient aux résultats obtenus par la réécriture de Q_2 à travers les registres de services possibles SR_1 et SR_2, i.e., les services impliqués dans C_1 (s_1, s_2) sont associés dans SR_1 et SR_2 à des vues sémantiques (v_{11}, v_{21}) dont la combinaison C_{1_intp} couvre Q_2. Par conséquent, la probabilité de l'interprétation C_{1_intp} est égale à $Pr(SR_1) + Pr(SR_2)$=0.35.

4 Evaluation efficace de requêtes

Dans le chapitre précédent, nous avons proposé un algorithme qui permet de calculer efficacement l'ensemble $Q^{rank}(PSR)$, et cela sans avoir besoin de générer l'ensemble des registres possibles $pwd(SR)$. Dans cette partie, nous appliquons presque les mêmes étapes de l'Algorithme 1, sauf la troisième partie, dans laquelle nous ne pouvons pas utiliser le **Théorème 1** pour calculer les probabilités des compositions obtenues. Ceci s'explique par le fait que le **Théorème 1** se base sur le modèle BID, où une interprétation de composition ne peut contenir que des vues sémantiques indépendantes. Cependant, dans notre cas, les vues sémantiques impliquées dans une interprétation sont souvent corrélées (ne sont pas

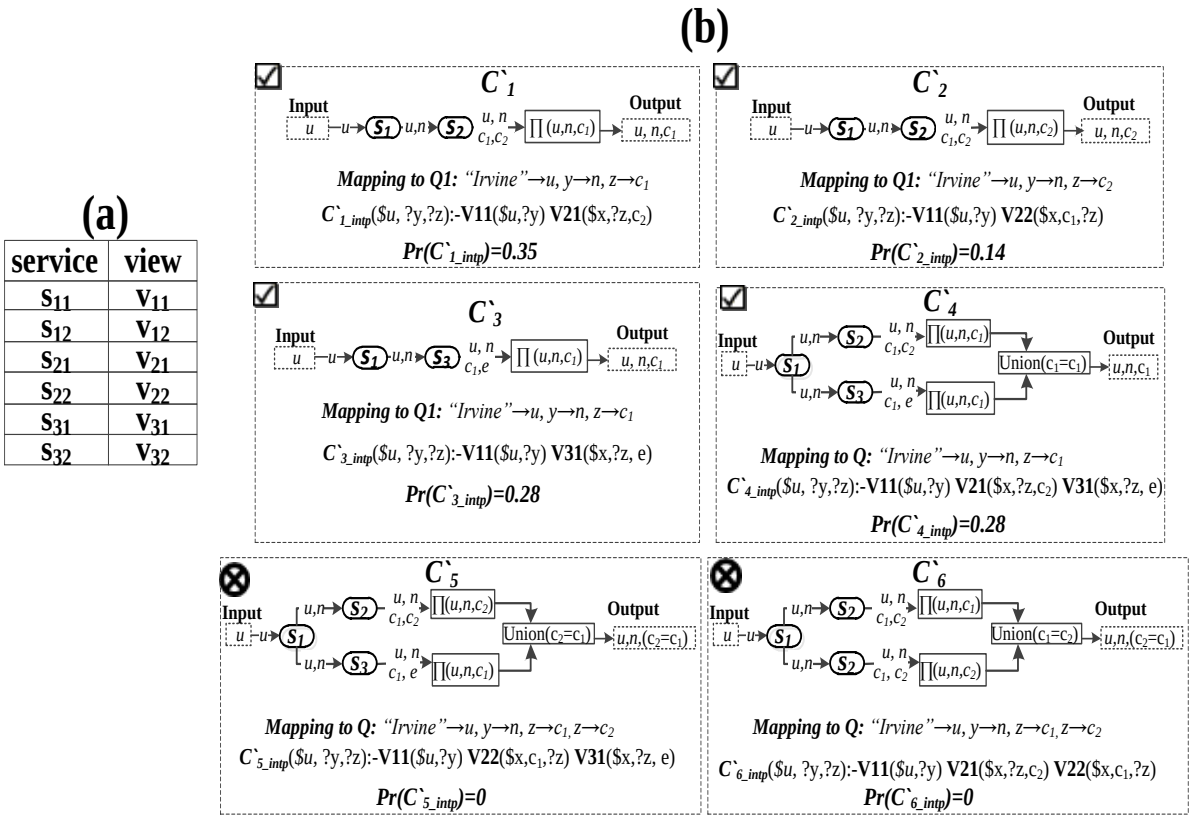


FIGURE 4.5 – (a)Registre de services déterministe DSR_2 de PSR_2.(b)Les compositions obtenues avec leurs probabilités.

indépendantes), ce qui contredit le **Théorème 1**.

Les deux premières étapes de l'Algorithme 1 permettent de réécrire une requête utilisateur Q directement sur PSR, tout en appliquant un processus de transformation sur ce dernier. Le résultat de cette réécriture est constitué d'un ensemble de compositions (virtuelles) $C'=\{C'_1, C'_2, ..., C'_m\}$, tel que chaque composition virtuelle $C'_h \in C'$ a une interprétation C'_{h_intp} qui couvre Q. Dans le reste de cette section, nous nous intéressons au calcul de la probabilité de chaque composition $C'_h \in C'$.

Example. Le registre de services probabiliste PSR_2 dans la figure 4.2(a) est transformé en un registre de services déterministe DSR_2 représenté dans La figure 4.5(a). La réécriture de la requête Q_2 (de la section précédente) sur DSR_2 permet de retourner six compositions (virtuelles) possibles (voir la figure 4.5(b)).

Calculer la probabilité d'une composition (virtuelle) C'_h revient à calculer la probabilité

de son interprétation C'_{h_intp}. Soit PSR un registre de services probabiliste qui est décrit par un réseau bayésien BN :(X, E, $\mathcal{P}$), tel que chaque vue sémantique v_{ij} (i.e., tuple $<s_i$, v_{ij}, $Pr(v_{ij})> \in PSR$) est associée à une variable aléatoire $x_{ij} \in X$. Par conséquent, C'_{h_intp} correspond dans BN à un sous-ensemble de variables aléatoires $X_{C'_h} \subseteq X$, tel que $X_{C'_h}$ contient uniquement les variables aléatoires qui sont associées avec les vues sémantiques (impliquées dans) de C'_{h_intp}. De plus, $X_{C'_h}$ est associée avec un assignement $AX_{C'_h}$, dans lequel chaque variable aléatoire $x_{ij} \in X_{C'_h}$ est assignée à 1. Pour calculer la probabilité de l'interprétation C'_{h_intp}, il suffit de calculer la probabilité marginale de $X_{C'_h}$ (ou $X_{C'_h} \sim AX_{C'_h}$). Formellement, la probabilité marginale de $X_{C'_h} \sim AX_{C'_h}$ est définie comme suit :

$$Pr(X_{C'_h}) = \sum_{AX_k \cong AX_{C'_h}} Pr_{BN}(X \sim AX_k),$$

où $\{AX_1, AX_2 \ldots, AX_n\}$ est l'ensemble de tous les assignements complets de X qui respectent l'assignement $AX_{C'_h}$.

Le calcul de la probabilité marginale nécessite un processus d'inférence dans le réseau bayésien BN. Généralement, la complexité d'une inférence probabiliste est NP-Hard. Cependant, plusieurs algorithmes ont été proposés afin d'optimiser cette inférence probabiliste [28, 55, 78]. Dans notre approche, nous adoptons l'algorithme *Variables Elimination* (VE) [28]. VE est un algorithme d'inférence probabiliste précis qui peut être utilisé pour calculer la probabilité marginale de tout sous-ensemble de X. l'avantage d'utiliser VE, est qu'il est actuellement l'algorithme le plus performant en terme de complexité [108].

Dans notre cas, pour calculer la probabilité de C'_{h_intp}, VE prend comme input : le réseau bayésien BN :(X, E, $\mathcal{P}$) de PSR, un sous-ensemble de variables aléatoires $X_{C'_h} \subseteq X$, et un ordre d'élimination $\mathcal{O}$ (i.e., Les variables éliminées sont celles qui n'appartiennent pas à $X_{C'_h}$), et retourne la probabilité marginale de $X_{C'_h}$. Cette probabilité est calculée à partir de BN en multipliant toutes les distributions de probabilités conditionnelles dans $\mathcal{P}$, et en sommant sur les variables éliminées (suivant l'ordre d'apparence dans $\mathcal{O}$).

Avant de calculer la probabilité marginale, VE supprime de BN toutes les variables aléatoires non- pertinentes. Une variable x_{ij} est non-pertinente *ssi* $x_{ij} \notin An(X_{C'_h})$, où $An(X_{C'_h})$ et l'ensemble d'ancêtres de $X_{C'_h}$. Pour supprimer x_{ij} de BN nous appliquons les étapes suivantes :

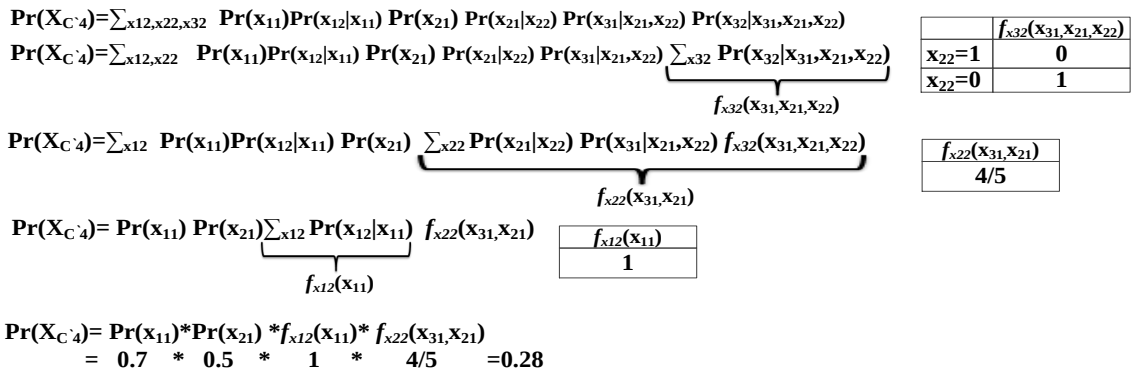


FIGURE 4.6 – La probabilité marginale de $Pr(X_{C'_4})$ en utilisant VE

– (i) supprimer x_{ij} de l'ensemble de variables aléatoires X,

– (ii) supprimer de E tous les arcs contenant x_{ij},

– (iii) supprimer de $\mathcal{P}$ toutes les distributions de probabilités conditionnelles impliquant la variable x_{ij}.

Example. Dans la figure 4.5(b), C'_{4_intp} implique les vues sémantiques v_{11}, v_{21} et v_{31}. En se basant sur le réseau bayésien BN_2 dans la figure 4.2(c), la probabilité de C'_{4_intp} est égale à la probabilité marginale de $X_{C'_4}$=$\{x_{11}, x_{21}, x_{31}\}$ (chaque variable dans $X_{C'_4}$ est assignée à 1). La figure 4.6 montre comment l'algorithme VE calcule la probabilité marginale de $X_{C'_4}$.

VE a besoin d'éliminer trois variables (i.e., x_{12}, x_{22}, and x_{32}, puisque ils ne font pas partie de $X_{C'_4}$) en multipliant six distributions de probabilités conditionnelles (i.e., $Pr(x_{11})Pr(x_{12}|x_{11})$,$Pr(x_{21})$ $Pr(x_{22}|x_{21})$, $Pr(x_{31}|x_{21},x_{22})$, $Pr(x_{32}|x_{31},x_{21},x_{22})$). Supposons que nous avons choisi l'ordre d'élimination suivant : $\mathcal{O}$=$\{x_{32}, x_{22}, x_{12}\}$.

Pour éliminer x_{32}, VE devrait d'abord sommer sur x_{32} à partir de la distribution $Pr(x_{32}|x_{31},x_{21},x_{22})$ pour produire une nouvelle distribution $f_{x_{32}}(x_{31},x_{21},x_{22})$ (l'indice inférieur indique la variable qui vient d'être éliminée en sommant sur elle). Pour éliminer x_{22}, VE devrait multiplier les distributions $Pr(x_{22}|x_{21})$ et $Pr(x_{31}|x_{21},x_{22})$, et puis sommer sur (éliminer) x_{22}, ce qui permet d'obtenir la distribution $f_{x_{22}}(x_{31}x_{21})$. Enfin, pour éliminer x_{12}, VE somme sur x_{12} en produisant $f_{x_{12}}(x_{11})$.

Une fois toutes les variables dans $\mathcal{O}$ sont éliminées, nous pouvons calculer la probabilité de $X_{C'_4}$, tel que $Pr(X_{C'_4})$=$Pr(x_{11})\times Pr(x_{21})\times f_{x_{12}}(x_{11})\times f_{x_{22}}(x_{31}x_{21})$=$0.7\times0.5\times1\times\frac{4}{5}$=0.28.

Certaines compositions peuvent avoir de fausses interprétations, i.e., ne satisfont pas les corrélations qui existent entre les vues sémantiques dans PSR. La probabilité de telles compositions est égale à zéro, ce qui les élimine de l'ensemble des compositions possibles $Q^{rank}(PSR)$. Par exemple, la probabilité de la composition $C_6^{'}$ est égale à 0, parce que son interprétation $C_{6\_intp}^{'}$ implique les vues sémantiques v_{21} et v_{22}, ce qui est contradictoire à la corrélation $(v_{21} \oplus v_{22})$. La composition $C_5^{'}$ est également éliminée, puisque son interprétation $C_{5\_intp}^{'}$ ne satisfait pas la corrélation $(v_{31} \rightarrow \neg v_{22})$.

L'algorithme 3 permet de générer l'ensemble de compositions possibles en réécrivant une requête utilisateur Q à travers un PSR corrélé, i.e., représenté par un réseau bayésien BN. Cet algorithme est divisé en 4 parties. La première partie (ligne 1) transforme le registre de services probabiliste PSR en un registre de services déterministe DSR associé avec une table de mapping SMT. Dans la deuxième partie (ligne 2), un algorithme de réécriture de requêtes QRA est utilisé pour générer les compositions candidates en reformulant Q via DSR. Dans la troisième partie (lignes 3-7), nous calculons la probabilité de chaque composition virtuelle $C_k^{'}$. Le calcul des probabilités est réalisé en 4 étapes : (i) la première étape (ligne 4) consiste à identifier à partir de BN le sous-ensemble des variables aléatoires $X_{C_k^{'}}$ qui correspond à l'interprétation $C_{k\_intp}^{'}$; (ii) La deuxième étape (ligne 5) permet de supprimer les variables aléatoires non-pertinentes tout en créant un nouveau réseau bayésien $\bar{BN}$ (un extension de BN) ; (iii) Dans la troisième étape (ligne 6), nous définissons de façon aléatoire l'ordre d'élimination $\mathcal{O}$; (iv) La quatrième étape (ligne 7) calcule la probabilité marginale de $X_{C_k^{'}}$ en utilisant l'algorithme d'inférence *Variable Elimination* (**VE**). Enfin, après avoir calculé les probabilités des compositions, la dernière partie (lignes 8-11) consiste à transformer chaque composition virtuelle $C_k^{'}$ en une composition exécutable C_k.

Notons que l'Algorithme 3 peut être étendu pour calculer efficacement l'ensemble des k meilleures compositions. Cette extension consiste à appliquer la même stratégie de l'Algorithme *Top-K* (l'algorithme 2 du chapitre 3) tout en utilisant l'algorithme *VE* pour calculer les probabilités des compositions.

Algorithm 3: Génération des compositions possibles en présence des corrélations

Input: Q : Requête, PSR :Registre de services probabiliste, BN :(X, E, $\mathcal{P}$) : Réseau bayésien.
Output: *Possible_compo* : Ensemble des compositions possibles.

```
(DSR, SMT) ⟵ Transformation(PSR) ; // Transformation de PSR
Virtual_compo ⟵ QRA(Q, DSR) ; // Réécriture de requête
for each C'_k ∈ Virtual_compo do
    X_{C'_k} ⟵ FindVariables(BN.X, C'_{k_intp}) ;
    B̄N ⟵ RemoveIrrelevantVariables(BN, X_{C'_k}) ;
    O ⟵ RandomOrder(B̄N.X - X_{C'_k}) ;
    C_{k_pr} ⟵ VE(B̄N, X_{C'_k}, O) ;
for each C'_k ∈ Virtual_compo do
    C_k ⟵ CompositionTransform(C'_k, SMT) ;
    C_{k_pr} ⟵ C'_{k_pr} ;
    Possible_compo ⟵ Possible_compo ∪ {<C_k, C_{k_pr}>} ;
return Possible_compo ;
```

5 Conclusion

Dans ce chapitre, nous avons présenté une nouvelle approche pour la gestion et l'interrogation des services dont les sémantiques sont corrélées. Dans notre approche, nous utilisons les réseaux bayésiens pour modéliser de façon explicite les corrélations qui existent entre les vues sémantiques. Les probabilités des compositions obtenues peuvent être calculées efficacement en se basant sur l'algorithme d'inférence probabiliste *Variables Elimination* (VE).

Chapitre 5

Implémentation et évaluation

Dans la première partie de ce chapitre, nous allons présenter les systèmes et les architectures qui implémentent nos différentes contributions. Dans la deuxième section, nous analysons la complexité des différents algorithmes proposés dans le cadre de cette thèse. La troisième partie de ce chapitre est consacrée aux expérimentations que nous avons réalisé et aux résultats obtenus.

1 Implémentation

Dans cette section, nous présentons deux systèmes de composition de services DaaS. Ces systèmes implémentent respectivement les approches proposées dans le chapitre 3 (i.e., composition des services à sémantique non-corrélée), et le chapitre 4 (i.e., composition des services à sémantique corrélée). Les deux systèmes ont été développés sous l'environnement de développement *JavaEE* v1.7 en utilisant comme éditeur *NetBeans* v8.0. Les services web sont déployés dans le serveur d'application *Glassfish* 4.0.

1.1 Système de composition de services DaaS à sémantique non corrélée

Afin de valider notre approche et d'évaluer la performance de nos différentes techniques proposées dans le chapitre 3, nous avons implémenté un système de composition de services DaaS dont l'architecture est illustrée dans la figure 5.1. Dans notre système, les services DaaS sont modélisés comme étant des vues RDF (requête SPARQL paramétrée) à travers une ontologie de domaine. Les services sont supposés indépendants (i.e., ils en-

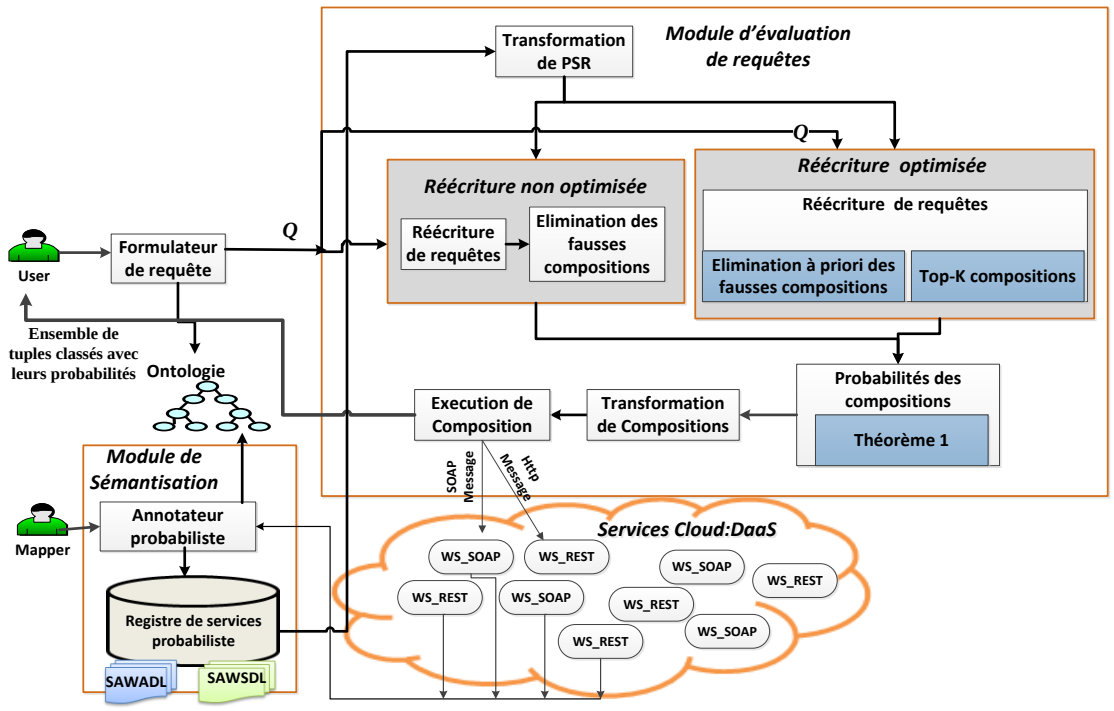


FIGURE 5.1 – Architecture du système de composition des services à sémantique non-corrélée

capsulent des sources de données différentes), et par conséquent les vues RDF qui sont associées à des services différents sont considérées comme étant des événements probabilistes indépendants. Autrement dit, les services ainsi que leurs vues RDF sont représentés par un registre de services probabiliste qui est définit par le ***Modèle BID***.

Les utilisateurs formulent leurs requêtes de composition (en SPARQL) à travers une ontologie de domaine. Le système résout ces requêtes en composant les services dont les sémantiques sont pertinentes. Notre système est constitué principalement de deux modules : *Module de sémantisation*, et *Module d'évaluation de requêtes*.

Module de sémantisation

Ce composant permet d'annoter la description de chaque service avec l'ensemble de ses vues RDF possibles (i.e., vues sémantiques). L'ensemble des vues RDF possibles peut être créé soit par un annotateur, soit en utilisant un outil de matching semi-automatique. Dans les deux cas, les sémantiques obtenues sont incertaines et probabilistes.

Les services DaaS peuvent appartenir à deux catégories de services, les services SOAP, et les services REST qui sont décrits respectivement par les descriptions WSDL et WADL. Ces descriptions peuvent être sémantiquement annotées en utilisant respectivement les langages d'annotation SAWDSL [65], et SAWADL [76, 77]. Ces langages utilisent les caractéristiques d'extensibilité de WSDL et WADL , tout en ajoutant des éléments et des attributs d'extension (e.g., *modelReference*, *liftingSchemaMapping*, etc). Dans notre système, nous étendons ces langages d'annotation en définissant un nouvel élément XML `Semantic-View` qui décrit une vue RDF possible. Cet élément `Semantic-View` est associé avec un attribut `Pr` qui spécifie la probabilité de la vue RDF correspondante. Un élément `Semantic-View` ne peut contenir qu'une seule vue RDF possible, i.e., le nombre des éléments `Semantic-View` ajoutés est égale au nombre des vues RDF possibles.

Example. Soit s_1 et s_2 des services de notre exemple de motivation (la section 1.1 du chapitre 3). Supposons que ces deux services sont respectivement de type SOAP, et REST. La figure 5.2(a) illustre une partie du fichier SAWSDL du service s_1. Dans ce fichier nous, avons ajouté à l'élément `Operation` deux éléments d'extension de type `Semantic-View`, puisque s_1 a deux vues sémantiques possible v_{11} et v_{12}, associées respectivement avec les probabilités 0.7 et 0.3. Le service s_2 a trois vues sémantiques v_{21} avec 0.3, v_{22} avec 0.6, et v_{23} avec 0.1. Par conséquent, le fichier SAWADL de s_2 doit contenir trois éléments de type `Semantic-View` ajoutés à la balise `method` (voir la figure 5.2(b)).

Module d'évaluation de requêtes

Ce module est constitué d'un ensemble de composants implémentant les différentes étapes de *l'Algorithme 1* (section 6 du chapitre 3), et cela afin d'assurer l'évaluation efficace des requêtes utilisateurs. Ce module implémente aussi les optimisations qui ont été proposées pour réduire l'effet des fausses compositions, ainsi que pour la génération efficace des Top-K compositions (section 7 du chapitre 3). L'avantage de notre approche est qu'elle peut être utilisée avec n'importe quel algorithme de réécriture certain. Dans notre système, le composant *Réécriture de requête* implémente l'algorithme de réécriture proposé dans [9].

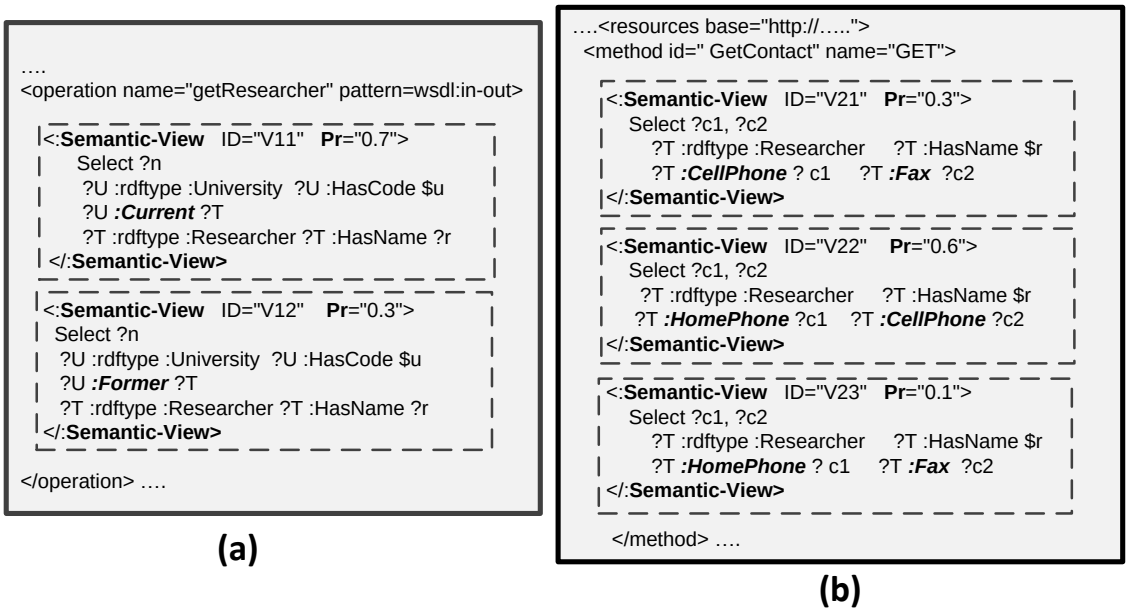


FIGURE 5.2 – **(a)** un fichier SAWSDL annoté avec deux vues RDF. **(b)** un fichier SAWADL annoté avec trois vues RDF.

1.2 Système de composition de services DaaS à sémantique corrélée

Dans cette deuxième implémentation, nous mettons en œuvre notre approche de représentation et de composition des services DaaS à sémantique corrélée (Chapitre 4). L'architecture du système implémenté est représentée dans la figure 5.3. Dans cette architecture nous réutilisons presque les mêmes composants de l'architecture précédente. Cependant, nous modifions et nous redéveloppons certains composants, tout en prenant en considération les corrélations qui existent entre l'ensemble des vues RDF. Par exemple, les composants *Elimination des fausses compositions* et *Elimination à priori des fausses compositions* ne figurent pas dans cette nouvelle architecture, parce que dans le cas des services à sémantique corrélée, le système ne peut éliminer les fausses compositions qu'après le calcul des probabilités. A la différence de l'architecture précédente, dans laquelle la probabilité des compositions est calculée en utilisant le **Théorème 1**, dans cette nouvelle architecture, le composant *Probabilités des compositions* calcule la probabilité des compositions de façon plus complexe en se basant sur l'algorithme d'inférence *Variable Elimination* (VE).

Les corrélations dans notre système sont gérées par le composant *Gestion des corrélations*. Ce composant s'ajoute au **Module de sémantisation**, et cela pour permettre la spécification

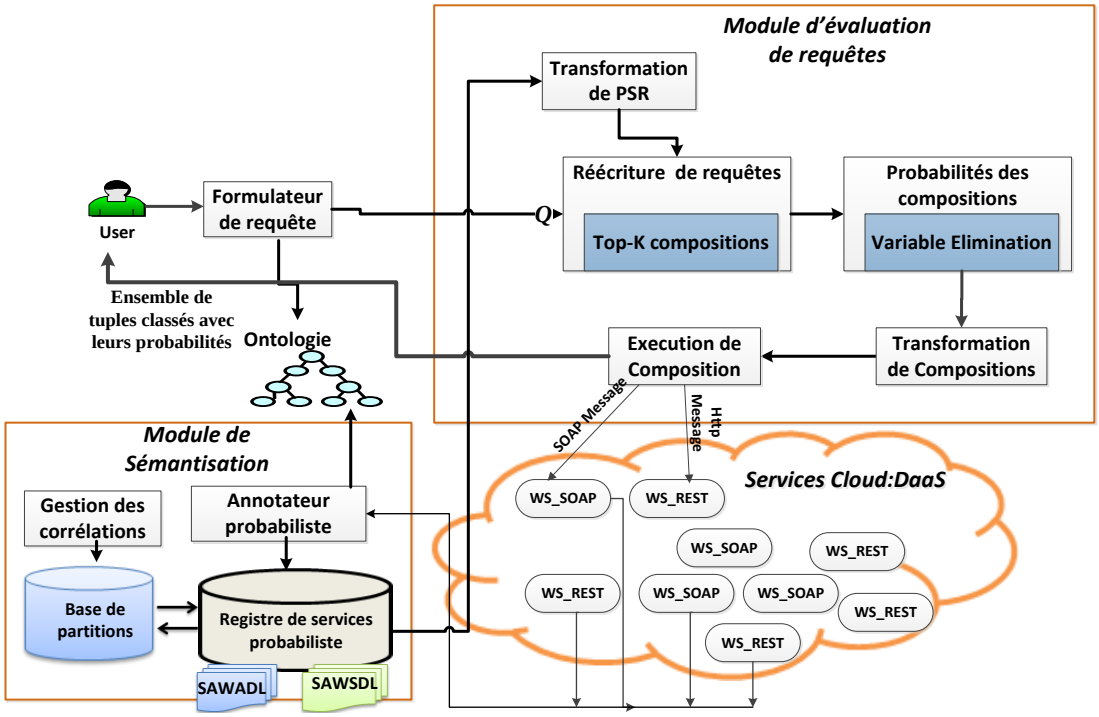


FIGURE 5.3 – Architecture du système de composition des services à sémantique corrélée

des corrélations qui existent entre les vues RDF des services DaaS. Dans notre système, nous avons implémenté certaines corrélations, e.g.,l' *Exclusion mutuelle* ($\oplus$), l'*Implication* ($\rightarrow$), l'*Equivalence* (xor), etc. Cependant, l'utilisateur peut spécifier d'autres corrélations plus complexes en définissant les tables de probabilités conditionnelles correspondantes. Notons que les vues RDF du même service sont par défaut disjointes, i.e., le système crée entre chaque deux vues RDF v_{ij} et v_{ik} associées avec le même service s_i une corrélation de type *Exclusion mutuelle*. Les vues entre lesquelles aucune corrélation n'est spécifiée sont traitées comme des événements indépendants.

Les services et leurs corrélations sont stockés séparément. Les services DaaS ainsi que leurs vues possibles sont décrits dans leurs descriptions SAWSDL et SAWADL. Pour stocker les corrélations, nous introduisons le concept de *partition*. Une partition est une table de probabilité conditionnelle associée avec un ensemble de références. Ces références décrivent les variables aléatoires des vues RDF corrélées. Les références sont représentées par une table relationnelle de quatre colonnes $<RefVue, URL, method, IdVue>$. La première colonne *RefVue* spécifie le nom de la variable aléatoires qui pointe vers une vue RDF

(a)

```
<operation name="getResearcher" pattern=wsdl:in-out>
  <:Semantic-View ID="V11" Pr="0.7" Ref="X11" >
     Select ?n
      ?U :rdftype :University  ?U :HasCode $u
      ?U :Current ?T
      ?T :rdftype :Researcher ?T :HasName ?n
  </:Semantic-View>
  <:Semantic-View ID="V12" Pr="0.3" Ref="X12" >
   Select ?n
   ?U :rdftype :University  ?U :HasCode $u
   ?U :Former ?T
   ?T :rdftype :Researcher ?T :HasName ?n
  </:Semantic-View>
</operation> ....
```

SAWSDL de S_1

```
....<resources base="http://.....">
 <method id=" GetContact" name="GET">
  <:Semantic-View ID="V21" Pr="0.5" Ref="X21" >
     Select ?c1, ?c2
       ?T :rdftype :Researcher     ?T :HasName $n
       ?T :CellPhone ? c1     ?T :HomePhone  ?c2
  </:Semantic-View>
  <:Semantic-View ID="V22" Pr="0.2" Ref="X22" >
     Select ?c1, ?c2
      ?T :rdftype :Researcher     ?T :HasName $n
      ?T :HomePhone ?c1    ?T :CellPhone ?c2
  </:Semantic-View>
   </method> ....
```

SAWADL de S_2

(b)

RefVue	URL	method	IdVue
x_{11}	Http://localhost/s1.sawsdl	getresearcher	V_{11}
x_{12}	Http://localhost/s1.sawsdl	getresearcher	V_{12}
x_{21}	Http://localhost/s2.sawadl	getcontact	V_{21}
x_{22}	Http://localhost/s2.sawadl	getcontact	V_{22}

refVue1	refVue2	PoValue1	PoValue2	ConditionalProb
x_{11}	x_{12}	1	0	1
x_{11}	x_{12}	1	1	0
x_{11}	x_{12}	0	0	0
x_{11}	x_{12}	0	1	1

(c)

refVue1	refVue2	PoValue1	PoValue2	ConditionalProb
x_{21}	x_{22}	1	0	1
x_{21}	x_{22}	1	1	0
x_{21}	x_{22}	0	0	3/5
x_{21}	x_{22}	0	1	2/5

(d)

FIGURE 5.4 – **(a)** Les fichiers SAWSDL et SAWADL de s_1 et s_2. **(b)** Table de références. **(c)** La partition qui correspond à la corrélation $v_{11} \oplus v_{12}$. **(d)** La partition qui correspond à la corrélation $v_{21} \oplus v_{22}$.

donnée ; La deuxième colonne *URL* et la troisième colonne *method* spécifient respectivement l'URL de fichier SAWSDL (ou SAWADL) et l'opération (ou method) de la vue RDF concernée ; La quatrième colonne *IdVue* décrit l'identifiant de la vue RDF choisie. Une partition est considérée comme étant une table relationnelle dont le schéma peut varier selon le nombre de variables aléatoires (vues RDF) impliquées dans la distribution de probabilités conditionnelles <*RefVue1*,...,*RefVueN*, *PoValue1*,...,*PoValueN*, *ConditionalProb*>. Les colonnes {*RefVue1*,...,*RefVueN*} représentent les variables aléatoires ; Les colonnes {*PoValue1*,...,*PoValueN*} spécifient les différentes valeurs possibles (0, ou 1) des variables aléatoires impliquées dans la partition ; Et enfin, la colonne *ConditionalProb* spécifie la probabilité conditionnelle qui est calculée en fonction des valeurs associées aux variables

aléatoires.

En outre, chaque vue RDF v_{ij} (i.e., un élément `Semantic-View` dans un fichier SAWSDL or SAWADL) contient une liste de pointeurs vers les partitions dont les références impliquent v_{ij}. Les services et les partitions forment ensemble une structure de données doublement chaînée.

Example. Soit s_1 et s_2 des services de l'exemple de motivation du chapitre 4. Supposons que ces deux services sont respectivement de type SOAP, et REST. La figure 5.4(a) illustre leurs fichiers SAWSDL et SAWADL. Le stockage des corrélations qui existent entre les vues RDF de s_1 et s_2 peut être réalisé en deux étapes. La première étape consiste à créer des références vers les vues RDF. La figure 5.4(b) présente un exemple d'une table de référence. Ensuite, la deuxième étape permet de créer l'ensemble de partitions (voir la figure 5.4(c)et (d)) en se basant sur les distributions de probabilités conditionnelles du réseau bayésien BN_2 dans la figure 4.2(c) (chapitre 4).

2 Analyse de la complexité des algorithmes proposés

2.1 Complexité de l'Algorithme 1

Dans cette partie, nous étudions la complexité de l'*Algorithme 1* (du chapitre 3) en analysant la complexité de ses différentes phases.

– Dans la première phase, chaque service incertain dans PSR est transformé en un ensemble de services virtuels dans DSR. Supposons que N est le nombre de services dans PSR (i.e., nombre de blocs), et V_{Max} est le nombre maximal de vues sémantiques possibles par service. La complexité de la première phase est $O(NV_{Max})$.

– Dans la deuxième phase, la requête utilisateur est réécrite sur DSR en utilisant un algorithme de réécriture certain. La complexité du problème de réécriture des requêtes est NP-complet [9] [95].

– Dans la troisième phase, l'interprétation de chaque composition est vérifiée pour voir si elle est fausse (i.e., implique des vues sémantiques disjointes). Supposons que L est le nombre de compositions, et V_{Max} est le nombre maximal de vues sémantiques par interprétation. La complexité de cette phase est $O(L{\times}V_{Max}{\times}V_{Max})$.

– Dans la quatrième étape, la probabilité de chaque composition est calculée en mul-

tipliant les probabilités des vues sémantiques impliquées dans son interprétation. Supposons que H est le nombre de compositions correctes, et V_{Max} est le nombre maximal de vues sémantiques par interprétation. La complexité de cette phase est $O(H \times V_{Max})$.

– Dans la dernière phase, les compositions virtuelles sont transformées en des compositions exécutables en se basant sur la table de de mapping SMT. Supposons que H est le nombre des compositions virtuelles, N est le nombre maximal de services par composition, et M est le nombre de tuples dans SMT. la complexité de cette phase est $O(H \times H \times N_{Max} \times M)$.

En résumé, les différentes phases de notre algorithme ont une complexité polynomiale, sauf la phase de réécriture de requêtes qui a généralement une complexité NP-complet. Par conséquent, la complexité de l'*Algorithme 1* dépend de celle de l'algorithme réécriture choisi. Les phases qui sont introduites pour gérer l'incertitude des sémantiques ont un coût très réduit par rapport à celui de la phase de réécriture, ce qui est un point fort de notre approche.

2.2 Complexité de l'Algorithme 3

L'*Algorithme 3* (du chapitre 4) est utilisé pour réécrire les requêtes à travers des services DaaS dont les sémantiques sont corrélées. L'*Algorithme 3* étend l'*Algorithme 1* en modifiant la quatrième phase (i.e., le calcul des probabilités des compositions). Dans ce cas, la probabilité des compositions est calculée en utilisant l'algorithme *Variable Elimination* (VE).

Dans notre cas, La complexité de VE est mesurée en fonction du nombre de vues sémantiques (variables aléatoires) impliquées dans l'interprétation d'une composition, et de la cardinalité des domaines (i.e., $dom(x_i)$) des variables aléatoires. Plus spécifiquement, la complexité de VE est égale à $O(2^{|X_{C'h}|})$, où $X_{C'_h}$ est le sous-ensemble de variables aléatoires correspondant à l'interprétation C'_{h_intp}, et 2 est la cardinalité des domaines des variables aléatoires booléennes (i.e., $dom(x_i)$=$\{0,1\} \rightarrow |dom(x_i)|$=2).

Cette complexité peut être réduite si nous appliquons un ordre d'élimination optimal. Cependant, trouver un ordre d'élimination optimal est un problème NP-complet.

2.3 Complexité de l'algorithme Top-k

L'algorithme *Top-k* (l'*Algorithme 2* du chapitre 3) s'ajoute aux algorithmes *Algorithme 1* et *Algorithme 3* pour optimiser la génération des k meilleures compositions. Cet algorithme étend la phase de réécriture (deuxième phase) en introduisant la notion de classe. Dans ce qui suit, nous analysons la complexité des étapes introduites par l'algorithme de *Top-k*.

– La première étape permet de construire l'ensemble des classes. Supposons que N est le nombre de services (virtuels) pertinents. La complexité de cette étape est O (N^2).

– La deuxième étape consiste à générer l'ensemble des *Top-k* classes. Cet ensemble est obtenu en classant les services (dans chaque classe) suivant leurs probabilités. Cette étape utilise l'algorithme de trie *Quick Sort* dont la complexité est $O(L\log_L)$, où L est le nombre de classes.

3 Expérimentation

Dans cette section, nous analysons les performances de notre approche en utilisant un registre de services probabiliste contenant 400 services DaaS générés aléatoirement, tel que chacun est associé avec 6 vues sémantiques possibles. Nous menons une série d'expérimentations qui s'adresse aux questions suivantes : (i) *Est-ce que nos algorithmes (Algorithme 1 et Algorithme 3) peuvent répondre aux requêtes dans un temps d'exécution raisonnable (comme dans le cas certain) ?* ; (ii) *Est-ce que les optimisations proposées permettent d'améliorer les performances de nos algorithmes ?*. Les expérimentations ont été réalisées dans une machine sous Windows dont les performances sont : 2.53 GHz Intel i3 core CPU, et 4 GB de RAM.

3.1 Analyse des performances de l'Algorithme 1

Le but de cette première expérimentation est de montrer que malgré la grande complexité de notre algorithme de réécriture incertain, les requêtes peuvent être encore résolues dans un temps raisonnable. En ce sens, nous comparons les performances de l'algorithme de réécriture certain [9] qui considère que des services à sémantique certaine (i.e., associés avec une seul vue sémantique), avec notre *Algorithme 1* tout en augmentant le nombre

de services de 100 à 400, et en variant le nombre de vues sémantiques par service de 2 à 6. Dans cette expérimentation, nous supposons que l'ensemble des services ainsi que leurs vues sémantiques possibles sont décrits par le modèle **BID** (i.e., pas de corrélations complexes entre les vues sémantiques). Aussi, nous considérons tous les types de requêtes : les *Chain query*, et les *Star query* [95] [9].

Dans le cas de *Chain query*, les requêtes et les vues sémantiques sont considérées comme étant des chaînes de prédicats (i.e., des triplés RDF), où l'ensemble des variables distinguées (liées aux paramètres d'entrées/sorties du service) est constitué uniquement des variables du premier et du dernier prédicat. Pour ce type de requêtes nous traitons deux cas : (*i*) les requête et les vues sémantiques avec une longueur équivalente de 21 prédicats (10 object properties liant 11 classe), et (*ii*) les vues sémantiques avec une longueur inferieur à celle de la requête (7 prédicats pour les vues et 21 pour les requêtes).

Dans le premier cas, une vue sémantique est pertinente seulement si elle est identique à la requête. Les résultats obtenus (Figure 5.5 (a)) montrent que l'*Algorithme 1* peut réécrire une requête à travers 400 services en moins de 5 secondes (comme l'algorithme de réécriture certain), et cela quelque soit le nombre de vues sémantiques possibles. Le coût introduit par le traitement de l'incertitude est insignifiant. Ceci est expliqué par le fait qu'une seule vue sémantique par service peut être identique à la requête (puisque les vues sémantiques possibles qui sont associées avec le même service sont différentes), et par conséquent l'augmentation du nombre des vues sémantiques n'a pas une influence significative sur le nombre de réécritures.

Pour le deuxième cas (Figure 5.5(b)), lorsque le nombre de vues sémantiques par service est 2, la différence de performance entre les algorithmes reste insignifiante ; Mais quand ce nombre augmente, le temps de réécriture de l'*Algorithme 1* augmente par rapport à celui du cas certain (i.e., un service est associée avec une seule vue sémantique), avec une différence entre 15 % à 30 %. Cependant, pour un ensemble de 400 services, où chacun est associé avec 6 vues sémantiques, l'*Algorithme 1* reste encore performant et scalable, tel que le temps nécessaire pour réécrire une requête ne dépasse pas 16 seconds. Rappelons que les vues sémantiques considérées dans cette expérimentation sont plus courtes par rapport à la requête. Dans une telle situation, le même service peut avoir plusieurs vues sémantiques qui sont pertinentes à la requête, ce qui permet d'augmenter le nombre de

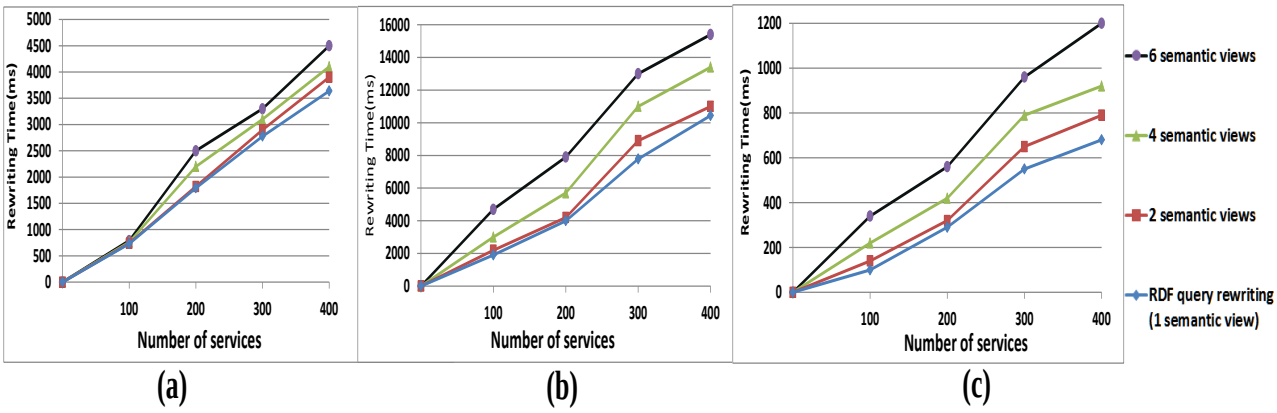


FIGURE 5.5 – (a) Chain requête/Vues avec 21 prédicats. (b) Chain requête avec 21 predicats et Chain vues sémantiques avec 7 prédicats. (c) start requête/vues avec 7 prédicats.

compositions, surtout celles qui sont fausses (souvent plus de 75 fausses compositions).

Dans le cas des requêtes/vues de type *Star query*, l'ensemble de tous les prédicats qui constitues une requête sont liés à un seul prédicat. La figure 5.5(c) montre les performance de notre algorithme pour traiter ce type de requête. Lorsque le nombre de vues sémantiques est inférieur ou égale à 4, le temps de réécriture des deux algorithmes (certain et incertain) ne dépasse pas une seconde. En général, le temps de réécrire n'est pas vraiment influencé par le nombre des vues sémantique, et il reste raisonnable dans tous les cas.

3.2 Effet de l'élimination à priori des fausses compositions

Dans cette section, nous évaluons l'effet de l'optimisation *élimination à priori des fausses compositions*. Nous commençons par l'évaluation du nombre de fausses compositions par rapport au nombre de services et au nombre de vues sémantiques par services. Les expérimentations dans les figure 5.6(a)(b) montrent que le nombre de services n'a pas d'influence sur le nombre des fausses compositions. Cependant, le nombre de fausses compositions augmente significativement lorsque le nombre de vues sémantiques dépasse 4, même si le nombre de services est petit, i.e., le nombre de fausses compositions est influencé par le degré de l'incertitude et non pas par le nombre de services. Ensuite, nous comparons les performances de l'*Algorithme 1* avec et sans optimisation. Dans cette expérimentation, nous utilisons les requêtes/vues de type *chain query*, avec 7 prédicats pour les vues sémantiques et 21 pour les requêtes, ce qui augmente la possibilité d'ob-

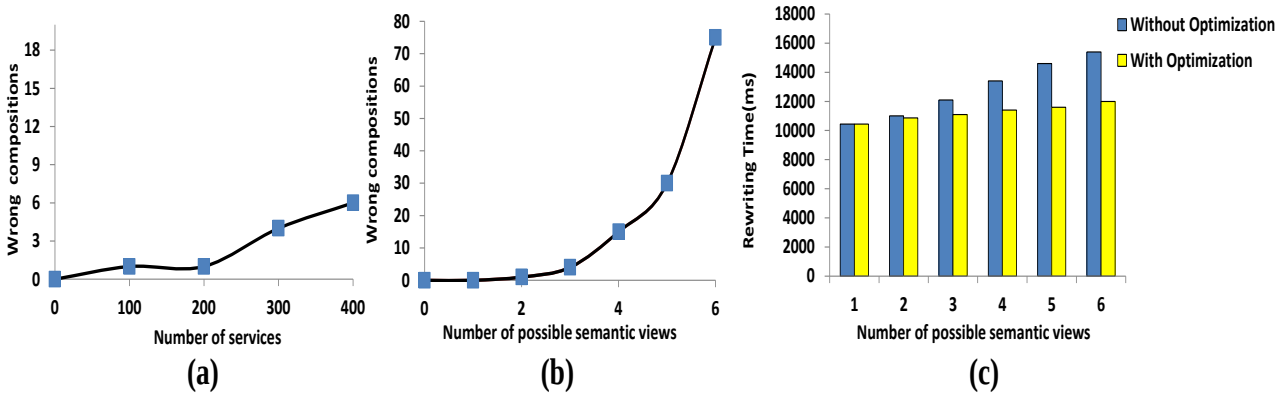


FIGURE 5.6 – (a) Fausses composition vs nombre de services (2 vues sémantiques par service) (b) Fausses composition vs nombre de vues sémantiques (nombre de services=100) . (c) Efficacité de l'*Elimination à priori des fausses compositions* (nombre de services=400).

tenir des fausses compositions. Les résultats obtenus dans la figure 5.6(c) montrent que notre optimisation réduit considérablement le temps de réécriture. Par exemple, lorsque le nombre vues sémantiques (par service) est 6, l'*Algorithme 1* réécrit une requête à travers 400 services dans un temps d'exécution qui s'approche de 16 secondes. Cependant, notre optimisation permet de réduire ce temps d'exécution à moins de 11 secondes. La raison derrière cette amélioration est que notre optimisation rend la deuxième phase de l'*Algorithme 1* en mesure d'éliminer toutes les fausses compositions avant d'appliquer le *Test de Couverture de Requête* qui est une tâche coûteuse en temps d'exécution.

3.3 Analyse des performances de l'Algorithme 3

Le but de cette expérimentation est d'analyser les performances de l'*Algorithme 3* en le comparant avec l'*Algorithme 1*. Pour cette expérimentation, nous utilisons deux registres de services probabilistes PSR_1 et PSR_2 qui contiennent chacun 400 services(6 vues sémantiques par service). Dans le premier registre PSR_1, les vues sémantiques qui sont associées à des services différents sont supposées indépendantes, et par conséquent PSR_1 est décrit par le modèle BID. En revanche, les services dans PSR_2 sont supposés dépendants (certains d'entre eux encapsulent les mêmes sources de données), ce qui permet de créer différentes corrélations complexes entre l'ensemble des vues sémantiques. Les corrélations dans PSR_2 sont représentées par un réseau bayésien. Pour les expérimentations nous uti-

lisons deux types de requêtes : *chain query* avec 21 prédicats pour les requêtes et les vues, et *chain query* avec 21 prédicats pour les requêtes et 7 pour les vues. L'*Algorithme 1* et l'*Algorithme 3* réécrivent ces requêtes en utilisant respectivement les registres PSR_1, et PSR_2. Les résultats obtenus sont représentés dans la figure 5.7. Pour le premier type de requêtes (requête et vues de même longueur 21 prédicats), nous ne remarquons pas une grande différence entre les deux algorithmes (voir figure 5.7(a)), e.g., pour les deux algorithmes une requête est réécrite à travers 400 services (6 vues sémantiques par service) en moins de 5 secondes. Ceci s'explique par le fait que les compositions obtenues ne peuvent contenir qu'un seul service (celui dont la vue sémantique est identique à la requête), et par conséquent inférer (à partir du réseau bayésien) la probabilité marginale de ces compositions revient à supprimer un nombre important de variables aléatoires non-pertinentes (i.e., le nombre de variables à éliminer est très réduit). Dans le cas où les vues ont une longueur inférieure à celle des requêtes, nous remarquons une différence de 2 à 6 secondes entre l'*Algorithme 1* et l'*Algorithme 3* (voir figure 5.7(b)). Dans ce cas, les compositions obtenues impliquent plusieurs services, ce qui complexifier le processus d'inférence (i.e., le nombre de variables à éliminer peut être important). Cependant, le temps de réécriture de l'*Algorithme 3* reste toujours raisonnable, e.g., Le temps nécessaire pour réécrire une requête à travers 400 services corrélés (6 vues sémantiques par service) est 22 secondes.

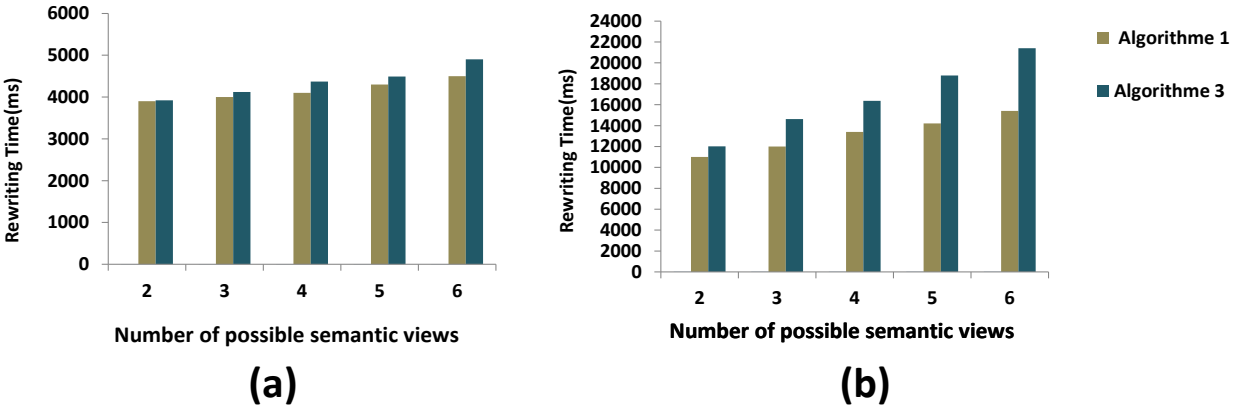


FIGURE 5.7 – (a) Algorithm1 vs Algorithme 3 (Chain requête/Vues avec 21 prédicats) (nombre de services=400). (b) Algorithm1 vs Algorithme 3 (Chain requête avec 21 prédicats et 7 pour les vues sémantiques) (nombre de services=400).

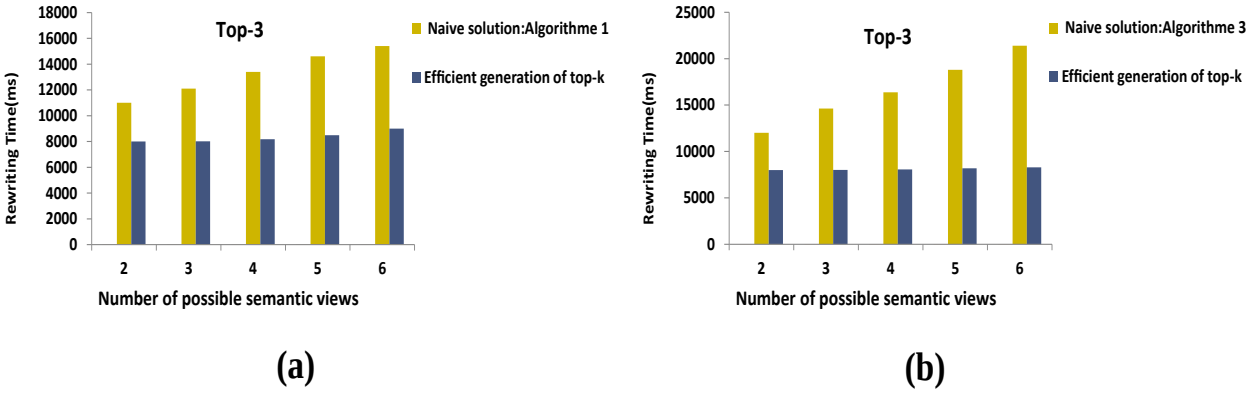


FIGURE 5.8 – (a) Algorithm Top-k vs Algorithme 1 (Chain requête/Vues avec 21 prédicats) (nombre de services=400). (b) Algorithm Top-k vs Algorithme 3 (Chain requête avec 21 prédicats et 7 pour les vues sémantiques) (nombre de services=400).

3.4 Effet de l'algorithme Top-k

Dans cette section, nous évaluons l'efficacité de notre algorithme *Top-k* (algorithme 2 du chapitre 3) en comparant ses performances avec l'*Algorithme 1* et l'*Algorithme 3*. Ces deux derniers algorithmes génèrent l'ensemble des *Top-k* compositions en utilisant une stratégie naïve, i.e., ils génèrent l'ensemble de toutes les compositions possibles, puis ils choisissent les k meilleures compositions. Dans cette expérimentation, nous utilisons les requêtes/vues de type *chain query*, avec 7 prédicats pour les vues sémantiques et 21 pour les requêtes. Les résultats obtenus dans la figure 5.8(a)(b) montrent que notre stratégie de *Top-k* réduit considérablement le temps nécessaire pour calculer les meilleures compositions. Par exemple, pour générer l'ensemble des *Top*-3 compositions (400 services, 6 vues sémantiques par chacun), le temps d'exécution de l'*Algorithme 1* peut aller jusqu'à 16 secondes (figure 5.8(a)). En revanche, lorsque nous appliquons l'algorithme de *Top-k*, les *Top*-3 compositions sont générées en moins de 8 secondes. De même pour l'*Algorithme 3*, la stratégie utilisée par l'algorithme de *Top-k* permet d'inférer la probabilité marginale de 3 compositions seulement, ce qui a un impact sur le temps de réécriture. Par exemple, en utilisant l'algorithme de *Top-k*, les *Top*-3 compositions sont générées à partir d'un registre de services probabiliste corrélé (400 services, 6 vues sémantiques par chacun) dans un temps d'exécution qui ne dépasse pas 9 secondes (figure 5.8(b)). Par compte, l'*Algorithme 3* génère le même ensemble de compositions avec un temps d'exécution de 22 secondes.

4 Conclusion

Dans ce chapitre, nous avons présenté les deux systèmes (à sémantique non-corrélée et à sémantique corrélée) que nous avons mis en oeuvre pour évaluer nos approches de composition de services incertains. Nous avons également effectué un ensemble d'expérimentation à travers un nombre important de services tout en évaluant l'efficacité et la faisabilité de nos solutions proposées. Les résultats obtenus montrent que nos différents algorithmes sont capables de réécrire des requêtes dans un temps raisonnable. De plus, les optimisations proposées réduisent considérablement le temps de réécriture.

Chapitre 6

Etat de l'art

1 Gestion des bases de données probabilistes

Actuellement, la plupart des applications produisant des données imprécises utilisent les modèles probabilistes (ou base de données probabiliste) pour quantifier l'incertitude des données. Dans la littérature, les travaux de recherches traitant les bases de données probabilistes s'intéressent à deux objectifs particuliers :(1) modélisation et représentation des bases de données probabilistes, et (2) l'évaluation efficace de requêtes.

1.1 Modèles de représentation

Un des modèles les plus utilisé pour la représentation des bases de données probabilistes est le modèle *Tuple-Indépendant* [25,42]. Dans ce modèle, les tuples sont considérés comme étant des événements probabilistes indépendants, i.e., il n'y a aucune corrélation entre eux. La probabilité jointe de n'importe quelle combinaison de tuples (i.e., une base de données possible) est calculée en multipliant les probabilités des tuples présents ainsi que les probabilités complémentaires des tuples absents. Formellement, soit $BD_K \in pwd(BD)$ une instance possible d'une base de données probabiliste BDP=$(pwd(BD), Pr)$ décrite par le modèle *Tuple-Indépendant*. La probabilité de BD_k est calculée comme suit :

$$Pr(BD_k) = \prod_{t_i \in BD_k} Pr(t_i) \prod_{t_i \notin BD_k} 1 - Pr(t_i)$$

Cependant, la forte indépendance entre les tuples n'est pas toujours le cas pour la plupart des applications produisant des données incertaines. Par exemple, les Sensors [29]

produisent des données entre lesquelles il peut y avoir différentes dépendances/corrélations, i.e., des dépendances en termes d'espace, de temps, etc. Ceci nécessite des modèles et des formalismes capables de représenter les corrélations qui peuvent exister entre les tuples d'une base de données probabiliste.

Le modèle *Bloc-Indépendant-Disjoint*(*BID*) [5,24] fournit une représentation simpliste pour les bases de données probabilistes avec la possibilité d'exprimer uniquement une simple corrélation qui est la disjonction ou l'exclusion mutuelle. Dans le modèle *BID*, les tuples sont partitionnés en un ensemble de blocs, tels que les tuples du même bloc sont considérés comme étant des événements probabilistes disjoints, alors que les tuples qui appartiennent à des blocs différents sont indépendants. Fondamentalement, chaque bloc décrit les différents alternatifs d'un tuple incertain, tout en gardant la même clé primaire. Le système TRIO [104] propose le modèle *x-tuples* qui est similaire au modèle *BID*. La structure et les probabilités des mondes possibles sont potentiellement influencées par les corrélations qui existent entre les tuples (i.e., la disjonction), où les bases de données possibles contenant des tuples disjoints (i.e.,appartiennent au même bloc) sont éliminées. La probabilité jointe d'une instance possible BD_k est un produit avec un seul facteur pour chaque bloc, tel que si BD_k implique un tuple t_i d'un bloc $\mathcal{B}$ alors le factuer est égale à $Pr(t_i)$, sinon il est égale à $1 - \sum_{t_i \in \mathcal{B}} Pr(t_i)$ (i.e., BD_k n'implique aucun tuple du bloc $\mathcal{B}$).

Les modèles présentés précédemment ne peuvent être utilisés que pour des cas particuliers exigeant une simple dépendance entre les tuples. Toutefois, en plus de l'indépendance et de la disjonction, il peut y avoir d'autres corrélations complexes entre les tuples à savoir : l'implication, l'équivalence (i.e., corrélation positive), la différence (corrélation négative), etc [123]. Les auteurs dans [107] proposent un modèle générique qui est capable de représenter n'importe quel type de corrélation. Dans ce modèle, une base de données probabiliste est formellement décrite par un graphe probabiliste orienté (i.e., réseau bayésien) [93], tel que chaque tuple t_i est associé à une variable aléatoire booléenne x_i qui décrit l'incertitude de son existence (i.e., x_i prend 1 si t_i existe, 0 sinon). Ces variables correspondent aux noeuds du réseau bayésien. Les corrélations qui existent entre les tuples sont quantifiées en ajoutant des arcs orientés entre les variables aléatoires correspondantes, et en maintenant chaque noeud (i.e., variable) avec une distribution de probabilité conditionnelle(i.e., facteur) qui spécifie comment la valeur d'un noeud dépend de celles

de ses prédécesseurs (i.e., ceux avec lesquelles il est en corrélation). Chaque instance possible BD_k est exprimée comme étant un assignement complet AX_k de toutes les variables aléatoires, tel que si un tuple t_i appartient à BD_k alors sa variable correspondante x_i est assignée à 1 dans AX_k, 0 sinon. La probabilité de BD_k est égale à la probabilité jointe de l'assignement AX_k et qui est calculée en multipliant les valeurs retournées par toutes les distributions de probabilité conditionnelles (sachant l'assignement AX_k).

Cependant, Les réseaux bayésiens ne permettent de représenter que des bases données probabilistes dont les corrélations sont orientées. Plusieurs approches [58, 59, 84] utilisent un autre type des graphes probabilistes appelé *Réseau de Markov* [23], et cela afin de représenter les bases de données probabilistes dont les corrélations entre les tuples sont non-orientées, i.e., la corrélation est plus symétrique, et sa direction n'a pas une influence sur son interprétation. Le travail présenté dans [108] propose une approche qui prend en considération les deux genres de corrélations : orientée et non-orientée.

Dans la littérature, il existe plusieurs autres formalismes conçus pour la représentation des bases de données probabilistes. Le travail dans [6,88] propose un modèle appelé *World-set decomposition* qui permet de représenter une base de données probabiliste via son ensemble des mondes possibles. Cette approche se base sur un processus de décomposition [119] qui permet de décomposer l'ensemble des instances possibles en un ensemble de relations tout en comblant la complexité liée au nombre exponentiel des mondes possibles (i.e., en termes de stockage et d'évaluation de requêtes). Dans leur forme probabiliste, ces décompositions peuvent être considérées comme étant des réseaux bayésiens à faible profondeur. Li et al [68] proposent le modèle *And/Xor Tree* qui généralise le modèle *BID* en ajoutant une nouvelle corrélation appelée la *coexistence*, i.e., la présence (resp. l'absence) d'un tuple implique la présence de l'autre (resp. l'absence). Les dépendances entre les tuples sont représentées de façon hiérarchique via un arbre.

1.2 Evaluation des requêtes

L'évaluation des requêtes dans les bases de données probabilistes est un des grands challenges qui intéressent actuellement la communauté scientifique. Evaluer une requête via l'ensemble des mondes possibles (i.e., *sémantique des requêtes* section 2.3, chapitre 2) est un problème **NP-hard**, ceci s'explique par le fait que le nombre des instances possibles

à travers lesquelles une requête est évaluée est exponentiel. Dans le but d'assurer une évaluation tractable des requêtes, il est nécessaire de trouver des techniques d'évaluation plus efficaces (i.e., optimales en termes de complexité), et dont le résultat est similaire à celui obtenu via l'ensemble des mondes possibles.

Les approches d'évaluation de requêtes à travers les bases de données probabilistes à simple corrélation (e.g., *Tuple-Independant*, *BID*, *And/Xor Tree*, etc) peuvent être divisées en deux catégories : *évaluation intensionnelle*, et *évaluation extensionnelle*.

L'approche intensionnelle est une méthode d'évaluation de requêtes [42] qui évalue ces dernières en termes d'événements complexes. Dans cette approche, chaque tuple t_i est associé à un événement probabiliste atomique $e(t_i)$. L'évaluation intensionnelle d'une requête Q est divisée en deux étapes : (i) premièrement, elle calcule pour chaque tuple possible (i.e., tuple retourné) son lineage λ qui décrit ses provenances. Le lineage d'un tuple est une formule propositionnelle (i.e., événement probabiliste complexe) qui est obtenue en se basant sur les événements des tuples initiaux et ceux des tuples intermédiaires, ainsi que les opérations algébriques du plan de la requête (e.g., projection, jointure, etc) ; (ii) La deuxième étape dans l'évaluation intensionnelle permet de calculer la probabilité des tuples possibles en se basant sur leur lineage λ, i.e., la probabilité d'une formule propositionnelle $P(\lambda)$. L'avantage de cette approche et qu'elle assure l'évaluation correcte de n'importe quelle requête (i.e., les résultats sont en accordance avec la sémantique de requêtes). Cependant, il est impraticable d'utiliser cette approche pour l'évaluation de requêtes, et cela pour deux raisons. La première est que selon le plan de la requête et la taille de la base de données probabiliste, le lineage λ d'un tuple possible peut devenir très large, ce qui augmente significativement la complexité d'exécution. La deuxième raison tient au fait que le calcul de la probabilité d'un lineage $P(\lambda)$ est un problème NP-complet [25]. Plusieurs approches ont été proposées afin d'optimiser la complexité liée au calcul de la probabilité d'une formule propositionnelle $Pr(\lambda)$. Les auteurs dans [41, 87] proposent une technique approximative qui permet de retourner pour $Pr(\lambda)$ un intervalle $[L, U]$ représentant la borne inférieure L et la borne supérieure U de la probabilité de λ, i.e., $L \leq Pr(\lambda) \leq U$. Ces techniques d'optimisation deviennent de plus en plus effectives dans le cas où l'évaluation d'une requête ne nécessite pas des probabilités exactes pour les tuples retournés (e.g., les requêtes *Top-k*).

Dans l'approche extensionnelle [25, 26, 96], le processus d'évaluation d'une requête Q est entièrement guidé par les expressions de cette dernière, où le calcul des probabilités est impliqué dans le plan de Q. Par conséquent, il n'est pas nécessaire de séparer l'exécution de la requête du calcul des probabilités, i.e., les probabilités des tuples finaux sont calculées dans le moteur de base de données pendant le traitement de la requête. Pour pouvoir calculer ces probabilités, l'approche extensionnelle étend les opérateurs algébriques (sélection σ, projection Π, jointure $\bowtie$, etc) du plan de Q en des opérateurs algébriques probabilistes (sélection probabiliste σ^p, projection probabiliste Π^p, jointure probabiliste $\bowtie^p$, etc). Ces opérateurs probabilistes s'appliquent uniquement sur des tuples indépendants. La sélection probabiliste σ^p agit comme σ, tel que les tuples sélectionnés gardent leurs propres probabilités (i.e., $Pr_{\sigma^p}(t)$ est égale à $Pr(t)$ si t est sélectionné, 0 sinon) ; La projection probabiliste Π^p calcule la probabilité d'un tuple t comme 1-(1-p_1)(1-p_2)...(1-p_n), où p_1, p_2,...,p_n sont les probabilités de tous les tuples dont la projection retourne t ; Tandis que dans la jointure probabiliste $\bowtie^p$, la probabilité de chaque tuple t obtenu par la jointure de deux tuples t_1 et t_2 (i.e., t=t_1 $\bowtie^p$ t_2) est égale à $p_1 \times p_2$.

En général, l'approche extensionnelle est considérée comment étant une solution efficace qui n'assure l'évaluation correcte que pour un type particulier de requêtes i.e., l'approche extensionnelle est approximative. Lorsque les probabilités des outputs sont calculées correctement (comme si nous avons évalué Q à travers les mondes possible), le plan de la requête est *safe*. La complexité des requêtes qui admettent des plans *safe* est polynomial. Cependant, pour certaines requêtes les probabilités obtenues sont incorrectes. Ceci s'explique par le fait que les plans de ces requêtes produisent des tuples intermédiaires qui ne sont pas indépendants, un tel plan est appelé *unsafe*. En revanche, il est possible de trouver pour ce type de requêtes des plans *safe*, et par conséquent assurer une évaluation correcte pour elles. Cependant, trouver pour une requête *unsafe* un plan *safe* est un problème NP-hard.

Lorsqu'une base de données probabiliste est décrite par un graphe probabiliste (i.e., corrélation entre les tuples), le processus d'évaluation d'une requête consiste à étendre ce graphe probabiliste tout en créant de nouvelles variables aléatoires. Ces nouvelles variables correspondent aux tuples intermédiaires ainsi qu'aux tuples résultant. Sen et al [107] définie une algèbre qui permet de modifier les opérateurs algébriques du plan de la requête

de façon à préserver les corrélations entre tous les tuples (i.e., tuples initiaux, tuples intermédiaires, tuples résultant). Les opérateurs modifiés permettent d'ajouter des facteurs (i.e., distributions de probabilités conditionnelles) au graphe probabiliste. Par exemple, pour l'opération projection nous ajoutons un facteur qui décrit la corrélation *or* ; La jointure se traduit par la corrélation *and*, etc. La dernière étape consiste à calculer les probabilités des tuples résultant, tout en basant sur le graphe probabiliste étendu, i.e., celui qui correspond au plan de la requête. La probabilité d'un tuple résultant est égale à la probabilité marginale de la variable aléatoire correspondante. Plusieurs algorithmes d'inférence peuvent être utilisés pour calculer efficacement la probabilité marginale dans un graphe probabiliste. Les auteurs dans [108] étendent l'algorithme d'inférence *Variable Elimination* [28] en introduisant le concept *Facteur partagé*. Ce concept consiste à fusionner les facteurs qui partagent entre eux certaines variables aléatoires, ce qui permet de réduire le nombre de facteurs dans le graphe probabiliste, et par conséquent optimiser le calcul des probabilités marginales.

Une autre façon pour assurer l'évaluation efficace des requêtes est d'utiliser les vues matérialisées. Dans les bases de données classiques, les vues matérialisées constituent un concept très puissant qui permet d'optimiser l'exécution des requêtes en se basant sur les résultats des requêtes précédentes [2, 50]. Dans le cas des bases de données probabilistes, les vues matérialisées peuvent avoir un impact considérable sur l'évaluation des requêtes [24, 99]. Par exemple, dans l'approche extensionnelle, l'évaluation d'une requête *unsafe* est un problème NP-Hard. En revanche, lorsque nous réécrivons cette même requête en termes de vues, elle devient *safe*, et par conséquent sa complexité d'évaluation est polynomial. Cependant, le problème majeur dans l'utilisation des vues matérialisées pour le traitement des requêtes est comment trouver, représenter, et utiliser les relations d'indépendance entre les tuples d'une vue. En général, les tuples dans une vue peuvent être corrélés de façon complexe. L'une des solutions pour résoudre ce problème est de stocker le lineage de chaque tuple, mais celle-ci rend l'évaluation de requêtes moins efficace tout en complexifiant la définition des vues par rapport à la requête.

1.3 Discussion

Comme il est expliqué précédemment, les principaux challenges des bases de données probabilistes concernent la modélisation et la représentation des données incertaines, ainsi que l'évaluation de requêtes. Pour les modèles de représentation, leur principale préoccupation est le degré de dépendance entre les tuples d'une base de données probabiliste (i.e., corrélation). Certains modèles supposent il n'y a aucune corrélation entre les tuples (e.g., *Tuple-Indépendant*), d'autres introduisent des corrélations simplistes (e.g., *BID*, *And/Xor Tree*). Cependant, il y a des modèles qui sont plus génériques et capables de représenter n'importe quelle corrélation complexe (e.g., les modèles à base de réseau bayésien, réseau de markov, etc).

Dans notre travail, nous traitons l'incertitude des services DaaS. Cette incertitude concerne la sémantique des services, i.e., à la différence des bases données probabilistes, les données retournées par un service sont certaines, mais leur interprétation est incertaine. Pour la représentation des services à sémantique incertaine, nous adaptons deux modèles probabilistes, notamment le modèle *BID* et le modèle à base de réseau bayésien. Le modèle *BID* est utilisé pour les services qui sont indépendants (i.e., la corrélation existe uniquement entre les vues sémantiques du même service). Cependant, les réseaux bayésiens sont utilisés dans le cas où les services DaaS partagent entre eux les mêmes sources de données (i.e., différentes corrélations complexes sont possibles).

Les approches d'évaluation de requêtes à travers les bases de données probabilistes peuvent être classifiées selon 3 critères : les corrélations entre les tuples, l'exactitude de résultats, et la complexité. L'approche intensionnelle et l'approche extensionnelle ne peuvent être appliquées que pour les bases de données probabilistes à simple corrélation, e.g., les bases de données probabilistes de type *Tuple-Indépendant*, *BID*, etc. Lorsqu'une base de données probabiliste est décrite par un graphe probabiliste (i.e., des corrélations complexes entre les tuples), l'évaluation de requête est réalisée de façon plus complexe (complexité exponentielle) en étendant le graphe associé. Le résultat obtenu par l'approche intensionnelle est toujours similaire à celui obtenu par la *sémantique des mondes possibles*, mais le calcul des probabilités des tuples résultants est un problème NP-complet. La complexité de l'approche extensionnelle est polynomiale. Cependant, cette approche n'assure l'exactitude des résultats que pour les requêtes *safe*. De plus, la génération des

plans *safe* pour les requêtes *unsafe* est un problème NP-hard. L'approche à base de vues matérialisées est une approche d'optimisation qui s'ajoute aux approches d'évaluation précédentes.

Dans notre travail, nous proposons deux sémantiques (i.e., *Sémantique des résultats possibles* et *Sémantique des compositions possibles*) pour l'évaluation de requêtes à travers les registres de services probabilistes. Ces sémantiques s'appliquent aussi bien sur les registres de services probabilistes à corrélation complexe (i.e., décrit via un réseau bayésien) que pour ceux à simple corrélation (i.e., décrit par le modèle BID). Pour la *Sémantique des compositions possibles*, nous proposons deux algorithmes pour l'évaluation efficace de requêtes (i.e., un pour le modèle BID, et un autre pour les PSR à base de réseau bayésien). Ces algorithmes sont complétement différents par rapport à ceux des bases de données probabilistes. Par exemple, nos algorithmes s'appliquent sur une seule table probabiliste (i.e., Le registre probabiliste PSR) qui contient seulement trois colonnes (i.e., *service*, *view*, *prob*). Par contre, une base de données probabiliste peut impliquer plusieurs tables avec des schémas variables. En outre, les approches d'évaluation dans les bases de données probabilistes proposent des extensions pour les différents opérateurs algébriques (e.g., sélection, projection jointure, etc). Dans notre travail, nous n'utilisons que l'opérateur de sélection qui permet de sélectionner à partir de la table PSR les services pertinents.

2 Systèmes d'intégration de données incertains

L'intégration des schémas est l'activité qui permet de fournir une représentation unifiée de plusieurs sources de données. Les problèmes fondamentaux de l'intégration des schémas sont : (i) le *matching des schémas*, i.e., l'identification des correspondances, ou des *mappings* entre les schémas des sources de données ; Et (ii) le *fusionnement des schémas*, i.e., la création d'un schéma unifié (i.e., schéma global ou médiateur) en se basant sur les mappings identifiés. Dans les approches classiques, les correspondances entre chaque deux schémas sont définies par un seul mapping (ils sont 100% certain de son exactitude). Cependant, il n'est pas toujours possible de définir avec certitude les mappings entre les sources de données. Ceci peut être dû à plusieurs raisons : (i) les données appartiennent souvent à un domaine différent de celui du mappeur (e.g., Biomédicale, économique, etc) ;

(*ii*) L'utilisation des outils semi-automatiques qui se basent généralement sur des techniques de similarité approximatives ; etc.

Dans cette section, nous présentons les principaux travaux qui ont été proposés dans le cadre des systèmes d'intégration de données incertains. Ces travaux peuvent être catégorisés en trois classes : (1) matching des schémas avec incertitude ; (2) Schémas médiateurs incertains ; Et (3) evaluation des requêtes à travers les mappings incertains.

2.1 Matching des schémas avec incertitude

Dans un système d'intégration de données, la phase de matching est celle où l'incertitude est souvent générée. Celle-ci permet de définir des mappings incertains entre les schémas des sources de données (i.e., plusieurs mappings alternatifs). L'incertitude des mappings peut être représentée à l'aide de deux méthodes : méthode quantitative et méthode qualitative. Dans l'approche quantitative, chaque mapping possible a une probabilité qui définit son degré de certitude ; Les méthodes qualitatives utilisent la logique floue et la théorie des possibilités afin de représenter les préférences de l'exactitude des mappings.

Magnani et al. [73] proposent une méthode pour la gestion de l'incertitude entre les schémas. Dans ce travail, les correspondances entre les schémas sont définies de façon globale à l'aide de six relations sémantiques (e.g., equivalence $\equiv$, subset-subsumption $\subset$, superset-subsumption $\supset$, intersection $\cap$, disjointness $\oplus$, incompatibility $\nsim$, etc). Plus précisément, le résultat du processus de matching définit l'ensemble des mappings possibles qui peuvent être existés entre les relations plutôt qu'entre les attributs. Magnani et al. [73] suggèrent que l'indication d'une probabilité exacte pour chaque mapping possible est une tâche difficile. Par conséquent, ils introduisent la notion de l'*Intervalle de Probabilité*, tel que chaque mapping possible est associée à un intervalle $[L, U]$ qui décrit sa probabilité minimale L, et sa probabilité maximale U.

Nottelmann et al. [85,86] proposent une extension probabiliste pour le modèle Datalog (Probabilistic Datalog, ou *pDatalog*) dans le but d'encoder les mappings incertains entre les schémas de sources de données. A la différence de l'approche précédente dans laquelle les mappings possibles sont définis entre les relations, dans le travail de Nottelmann et al. [85, 86], les mappings sont définis au niveau des attributs. De plus, ces mappings sont de type

binaire [1] (i.e., *similaire* , *non similaire*). Dans le modèle *pDatalog*, un mapping possible est décrit par une règle : $\alpha\ S_1(a) \longleftarrow S_2(b)$, où α est la probabilité pour que l'attribut a dans le schéma S_1 soit similaire à l'attribut b dans le schéma S_2. La probabilité de chaque règle est calculée en agrégeant les résultats obtenus par un ensemble de classificateurs. Deux catégories de classificateurs sont utilisées : classificateurs à base de schéma (e.g., noms et types des attributs), et classificateurs à base de contenu (e.g., KNN, naive bayes, KL-distance).

Les auteurs dans [32] introduisent un framwork générique qui permet la génération des Top-k mappings en se basant sur un ensemble de systèmes de matching. Les similarités entre les attributs sont calculées par cet ensemble de systèmes de matching, et combinées pour générer le classement final. Par exemple, un ensemble de systèmes de matching peut impliquer des techniques de similarité à base de domaine, à base de nom, etc. Chaque système de matching utilise une fonction d'agrégation locale (e.g., moyenne) pour générer la mesure de similarité des schémas à partir des mesures de similarité des attributs. La génération des Top-k mappings est réalisée de façon générique en utilisant les mesures de similarité obtenues. Le travail dans [32] support plusieurs algorithmes pour la génération des meilleurs mappings, à savoir : **Threshold** [38], **Matrix-Direct** (une simple extension de l'algorithme COMA [31]), **CrossThreshold** (une solution hybride des deux derniers algorithmes).

Dong et al. [33, 34] proposent une première analyse formelle des mappings incertains. Dans leur travail, les correspondances entre un schéma médiateur T et un des schémas sources S sont représentées par un mapping probabiliste PM :(T, S, m), où m :{(m_1, $Pr(m_1)$), (m_2, $Pr(m_2)$),. . . , (m_n, $Pr(m_n)$)} est l'ensemble des mappings possibles. Chaque mapping possible m_i est associé à une probabilité $Pr(m_i)$, i.e., PM définit une distribution de probabilité sur l'ensemble des mappings possibles. La somme des probabilités de tous les mappings possibles est égale à 1. Les mappings probabilistes peuvent être interprétés de deux manières : *by-table*, et *by-tuple*. Dans l'interprétation *by-table*, un seul mapping dans PM est correcte, et il est appliqué sur tous les tuples dans S. En revanche, dans l'interprétation *by-tuple*, plusieurs mappings sont partiellement correctes, et chacun est approprié pour un sous-ensemble de tuples dans S, par conséquent il n'est pas possible de

1. La plupart des travaux utilisent ce type de relation pour représenter les mappings entre les schémas

spécifier pour tout tuple le mapping le plus convenable.

D'après Dong et al. [33,34], le nombre des mappings possibles peut être très large voire exponentiel, surtout dans le cas où le mapping probabiliste correspondant est interprété par la sémantique *by-tuple*. Dans ce sens, ils proposent trois représentations permettant de réduire considérablement la taille des mappings probabilistes. La première représentation permet de partitionner l'ensemble de tous les attributs (i.e., les attributs dans T et S) en groupes, et ensuite les mappings possibles sont spécifiés pour chaque groupe séparément. La deuxième représentation permet de représenter un mapping probabiliste par les probabilités marginales des correspondances entre les attributs. La dernière représentation consiste à décrire les mappings probabilistes via les réseaux bayésiens (ou graphe probabiliste orienté). Ces réseaux bayésiens permet de réduire la complexité de stockage des mappings possibles de $O(2^n)$ à $O(n)$.

Une approche intéressante [27, 105] a été proposée pour adresser le problème de la génération des mappings probabilistes dans les systèmes d'intégration de données automatiques (i.e., une intégration à la demande). La première étape consiste à calculer l'ensemble des correspondances pondérées CP entre les attributs du schéma médiateur T et les attributs du schéma source S. Une correspondance pondérée $CP_{ij} \in CP$ définit une similarité agrégée entre le i-ème attribut dans T et le j-ème attributs dans S. Plus précisément, $CP_{ij}=\left[\sum_{k=1}^{l} sim_k(a_i, b_j)\right]/l$, où sim_k est la k-éme méthode de matching utilisée pour calculer le degré de similarité entre a_i et b_j. La deuxième étape permet de générer le mapping probabiliste PM en se basant sur l'ensemble des correspondances pondérées CP. Cependant, plusieurs mappings probabilistes peuvent être consistants avec le même ensemble des correspondances pondérées CP. Pour résoudre ce problème, Sarma et al. [27, 105] utilise l'*Entropie de Shannon* pour choisir le meilleur mapping probabiliste parmi ceux qui sont consistants avec CP. Le meilleur mapping probabiliste est celui dont la distribution de probabilité maximise l'*Entropie de Shannon*, i.e., maximise la valeur de $\sum_{j=1}^{n} -p_{ij}$* log p_{ij}, où $\{p_{i1}, p_{i2}, \ldots, p_{in}\}$ sont les probabilités des mappings possibles.

Une autre façon pour représenter les mappings incertains est d'utiliser les ensembles flous pour formaliser les similarités entre les attributs [43, 44], tel que les relations entre deux schémas S_1 et S_2 sont définies par une fonction $\mu : S_1 \times S_2 \longrightarrow [0,1]$.

2.2 Schémas médiateurs incertains

Un schéma médiateur est un ensemble de termes (e.g., relation, attributs) à travers lesquels les requêtes utilisateurs sont formulées. Dans les systèmes d'intégration de données de type GAV (i.e., Global as View), le schéma médiateur est défini à partir des schémas des sources locales. La génération du schéma médiateur se base souvent sur des outils semi-automatiques, ce qui rend celui-ci incertain. Dans la littérature plusieurs approches [20, 27, 51, 73, 105] ont adressé le problème de l'incertitude au niveau des schémas médiations, en proposant des modèles probabilistes.

He et al. [51] considèrent le problème de la génération d'un schéma médiateur pour un ensemble de sources Web. Le principe de base de cette approche est de créer un schéma global qui est statistiquement consistant avec les schémas sources. Pour ce faire, ils supposent que les schémas sources sont créés par un *modèle génératif* appliqué à certains schémas médiateurs, ce qui peut être considéré comme étant un schéma médiateur probabiliste. D'autres travaux [52, 53] proposent des modèles plus complexes et plus génériques en traitant les corrélations (i.e., dépendances probabilistes) qui peuvent être existées entre les schémas.

Les auteurs dans [72] proposent une solution non-probabiliste (à base de préférence) pour la génération de schéma médiateur. Si entre deux schémas d'objets S_1 et S_2 il n'y a aucune correspondance, ces schémas seront insérés dans le schéma médiateur comme deux entités différentes (i.e., des objets séparés). En revanche, s'il existe des similarités entre deux schémas d'objets S_1 et S_2, un troisième objet S_{12} sera aussi ajouté au schéma médiateur. Le nouveau schéma S_{12} applique les instances des schémas locaux S_1 et S_2, i.e., D_{12}=$D_1 \cup D_2$, où D_i est l'ensemble des données décrit par le schéma S_i. Ceci s'explique par le fait que les schémas locaux représentent une vue incomplète sur l'ensemble des données.

Sarma et al. [27,105] proposent un modèle probabiliste pour la génération et la représentation des schémas médiateurs incertains. Dans ce travail, l'interface unifiée d'un ensemble de schémas sources $\{S_1, S_2, \ldots, S_n\}$ est décrite par un schéma médiateur probabiliste PMS. PMS est constitué d'un ensembles de schéma médiateurs possibles$\{(ms_1, Pr(ms_1)), (ms_2, Pr(ms_2)), \ldots, (ms_m, Pr(ms_m))\}$, chacun avec une probabilité qui indique son degré de certitude, i.e., est ce qu'un schéma médiateur ms_i décrit correctement le domaine des

sources locales. L'ensemble des schémas médiateurs possibles est généré en appliquant un processus de clustering (i.e., regroupement) sur l'ensemble de tous les attributs des schémas sources. Chaque schéma médiateur possible est constitué d'un ensemble de clusters, tel que chacun implique des attributs similaires. Par exemple, l'attribut *address* dans une source S_1, et l'attribut *home-address* dans une autre source S_2 seront impliqués dans le même cluster (i.e., ils seront représentés par le même attribut dans le schéma médiateur). La probabilité d'un schéma médiateur possible ms_i est égale à la proportion du nombre de sources locales avec lesquelles il est consistant. Un schéma médiateur ms_i est consistant avec une source locale S_j, s'il n'existe pas un couple d'attributs dans S_j qui appartiennent au même cluster dans ms_i.

2.3 Evaluation des requêtes à travers les mapping incertains

Dans le cas certain, évaluer une requête Q dans un système d'intégration de données revient à la reformuler en une autre requête $Q^{'}$, et puis exécuter cette dernière sur les sources locales. Le processus de la reformulation des requêtes se base sur les mappings qui sont décrits entre le schéma médiateur et les schémas sources. Lorsque les mappings sont incertains, la reformulation des requêtes devient aussi incertaine, et plusieurs solutions alternatives sont possibles. Ci-dessous, nous décrivons les principaux travaux qui ont examiné le problème de la reformulation de requêtes à travers les schémas incertains.

Dong et al. [33, 34] étudient le problème de l'évaluation de requêtes SPJ (i.e., Select-Project-Join) à travers les mappings probabilistes, où chaque requête utilisateur Q est reformulée en un ensemble de requêtes possibles $\{Q^{'}_1, Q^{'}_2, \ldots, Q^{'}_n\}$. Ces requêtes possibles sont obtenues en se basant sur le mapping probabiliste $PM : \{(m_1, Pr(m_1)), (m_2, Pr(m_2)), \ldots, (m_n, Pr(m_n))\}$. Dans ce cas, chaque requête possible $Q^{'}_i$ est le résultat de la reformulation de Q à travers le mapping possible $m_i \in PM$. Le résultat final est l'agrégation de tous les tuples obtenus par l'exécution des requêtes possibles. Les auteurs dans [33, 34] montrent que lorsque les mappings probabilistes sont interprétés par la sémantique *by-table*, la complexité de l'évaluation de requêtes est polynomiale. Cependant, dans le cas de la sémantique *by-tuple*, cette complexité devient NP-complet. Les auteurs proposent aussi un algorithme de *Top-k* afin d'optimiser l'évaluation de requêtes, où k est le nombre de tuples à retourner. L'idée de base de cet algorithme est de réduire le nombre des requêtes

possibles à exécuter.

Gal et al. [45] étendent le travail de Dong et al. [33, 34] en traitant les requêtes d'agrégation, i.e., les requêtes qui impliquent des opérateurs d'agrégation : COUNT, MIN, MAX, SUM, AVG. Dans ce travail, le résultat de l'évaluation d'une requête d'agrégation Q est interprété de trois manière (i.e., trois sémantiques) :*range*, *expected-value*, *distribution*. La sémantique *range* retourne un intervalle qui décrit la valeur minimale et la valeur maximale de l'ensemble des résultats possibles. La sémantique *expected-value* retourne la valeur attendue (une seule valeur) de l'agrégat. La sémantique *distribution* retourne tous les résultats possibles avec leurs probabilités. Notons que les résultats des deux premières sémantiques peuvent être dérivés à partir de ceux de la dernière sémantique. Autrement dit, la sémantique *distribution* est la sémantique la plus exhaustive. Les auteurs étudient ces trois sémantiques à travers les deux interprétations des mappings probabilistes (*by-table*, *by-tuple*), et proposent des algorithmes polynomiales pour la plupart des opérateurs d'agrégation.

Les auteurs dans [19] proposent une solution efficace pour l'évaluation des requêtes à travers les mappings probabilistes. Dans cette approche, deux techniques d'optimisation sont développées : *q-sharing*, et *o-sharing*. Ces optimisations exploitent les similarités qui existent entre les mappings possibles, tout en réduisant les coûts de la *réécriture* et l'*exécution* des requêtes. L'algorithme *q-sharing* permet d'identifier les groupes de mappings qui réécrivent la requête utilisateur Q de la même façon, i.e., deux mappings possibles m_i m_j appartiennent au même groupe, si le résultat de la reformulation de Q à travers m_i et m_j est égale à la même requête possible Q'_k. Ceci permet de réduire le nombre de mappings possibles, et par conséquent le nombre de requêtes possibles à exécuter. Le deuxième algorithme *o-sharing* exploite la similarité entre les mappings au niveau des opérateurs (i.e., opérateurs impliqués dans les plans des requêtes possibles). Cette solution est destinée aux mappings qui ne sont pas complétement similaires. Un algorithme de *Top-k* est aussi proposé dans le cadre de ce travail. Cet algorithme utilise l'algorithme *o-sharing* pour calculer l'ensemble de toutes les réponses potentielles, ensuite il retourne les k meilleurs tuples en se basant sur leurs probabilités. Cependant, l'inconvénient de cette solution est que les probabilités des tuples sont approximatives. En outre, si le nombre des réponses potentielles est important, cet algorithme de *top-k* perd son efficacité.

2.4 Discussion

Comme nous le montrons dans cette section, les modèles probabilistes sont les plus utilisés dans les systèmes d'intégration de données. Les approches qualitatives sont généralement utilisées dans le but de réduire la complexité de la manipulation de l'incertitude.

Les approches de matching incertain peuvent être catégorisées selon quatre critères : le modèle de données, le type de relations entre les schémas, le modèle probabiliste, et l'interprétation des mapping. Le tableau 6.1 représente un récapitulatif sur les principales approches traitant l'incertitude des systèmes de matching.

TABLE 6.1 – Synthèse des approches de matching incertain

	Modèle	**Type de relation**	**Modèle probabiliste**	**Interprétation**
Magnani et al. [73]	Schéma d'objets	$\equiv$, $\subset$, $\supset$, $\cap$, $\oplus$, $\nsim$	Intervalle de probabilité	*by-table*
Nottelmann et al. [85]	Schéma d'objets	*similaire*, *non similaire*	p*Datalog*	*by-table*
Dong et al. [34]	Relationel	*similaire*, *non similaire*	Mapping probabiliste	*by-table*, *by-tuple*
Sarma et al. [105]	Relationel	*similaire*, *non similaire*	Mapping probabiliste, *Entropie de Shannon*	*by-table*

Dans notre travail, le schéma médiateur et l'ensemble des schémas sources sont respectivement représentés par une ontologie médiatrice et un ensemble de services DaaS. La correspondance (ou mapping) entre l'ontologie médiatrice et chaque service DaaS est décrite par une vue sémantique. Le type d'incertitude que nous traitons dans cette thèse concerne uniquement les vues sémantiques. L'ontologie médiatrice à travers laquelle les requêtes utilisateurs sont définies est supposée certaine. Pour la génération des vues sémantiques probabilistes d'un service donné, nous proposons un algorithme qui est inspiré de l'approche de Sarma et al. [27, 105]. Pour l'interprétation des vues sémantiques, notre travail ne prend en considération que l'interprétation *by-table*. L'ensemble des services ainsi que leurs vues sémantiques possibles sont représentées via un modèle robuste (i.e., registre de services probabiliste PSR) qui se base sur les principes des bases de données probabilistes.

Dans le contexte des mappings incertains, les approches d'évaluation de requêtes se distinguent selon le type de requêtes (i.e., requêtes SPJ, ou requêtes d'agrégation), et selon l'interprétation des mappings probabilistes (i.e., *by-table*, *by-tuple*). Dans notre thèse, nous traitons l'évaluation de requêtes SPARQL de type SPJ avec des vues sémantiques in-

terprétées selon la sémantique *by-table*. Toutefois, dans notre approche, nous nous intéressons beaucoup plus par les dépendances probabilistes (corrélations) qui existent entre l'ensemble des vues sémantiques possibles. De plus, nous proposons un algorithme de *Top-k* qui permet d'optimiser la génération des meilleures compositions.

3 Découverte et Composition de services web

La composition de services peut être définie comme le processus de découverte, et de combinaison de services en vue d'accomplir un objectif donné. La découverte de services est le processus qui permet de trouver l'ensemble des services pertinents en se basant sur leurs capacités. La combinaison de services décrit une interaction entre deux ou plusieurs services pertinents tout en satisfaisant la requête utilisateur.

Dans ce qui suit nous nous intéressons uniquement aux travaux qui traitent les services cloud de type SaaS et DaaS.

3.1 Découverte de services web

En générale, le processus de découverte se base principalement sur les capacités des services. La capacité d'un service est une représentation générique des fonctionnalités fournies par celui-ci. Dans la plupart des travaux proposés, les capacités des services sont définies par une description sémantique. Le processus de découverte nécessite aussi un processus de matching qui permet de comparer les capacités des services avec celles de la requête utilisateur.

Description sémantique des services web

Le standard WSDL présente des lacunes de précision dans la description d'un service. Ces lacunes sont liées au niveau d'expressivité faible de la description syntaxique car elle est limitée à l'énumération des opérations et à la description des types des paramètres d'entrée/sortie. Elle ne caractérise pas la sémantique de la fonctionnalité accomplie par le service. Pour pallier le manque de sémantique de WSDL, plusieurs approches proposent de rajouter une couche au-dessus de WSDL complétant la description syntaxique par des précisions sémantiques. La sémantique d'un service permet d'automatiser les tâches liées

à sa réutilisation, à savoir : la découverte, la composition, l'invocation, etc.

OWL-S (OWL based on ontology language for services) [7] est une ontologie particulière à base de OWL [2]. Elle est conçue pour décrire sémantiquement les capacités des services. L'ontologie OWL-S décrit un service à l'aide de trois classes : *Service Profile*, *Service Model*, et *Service Grounding*. Le *Service Profile* fournit une description de haut niveau d'un service et de son fournisseur. il implique trois types d'information : (i) une description du fournisseur de service, (ii) le comportement fonctionnel du service, et (iii) les propriétés non-fonctionnelles du services (e.g., QoS). Le comportement fonctionnel du service est défini par les propriétés **IOPE** i.e., Input, Output, Precondition, Effect. Le *Service Model* définit plusieurs types de processus (i.e., processus atomiques, processus simples, et processus composite) décrivant l'ordonnancement des opérations et le comportement d'un service dans une interaction. Le *Service Grounding* décrit les modalités de communication avec un service en précisant le protocole, le format des messages, la sérialisation et l'adressage. Il est le résultat d'une correspondance entre le *Service Model* décrivant le service et la description WSDL originale.

WSMO (Web Service Modeling Ontology) [100] fournie un modèle conceptuel pour la description sémantique des services dans le but d'automatiser leur découverte, leur composition, etc. WSMO est constitué de quatre éléments : (1) l'ontologie qui formalisme un domaine de connaissances ; (2) Les buts qui décrivent les objectifs des utilisateurs ; (3) Les services ainsi que leurs descriptions sémantiques ; (4) et des médiateurs qui assurent l'interopérabilité et résolvent les problèmes d'hétérogénéités entre les services. La description des éléments de WSMO est représentée en utilisant WSML (Web Service Modeling Language) [39].

SAWSDL(Semantic Annotation for WSDL) [65] présente un mécanisme permettant d'annoter sémantiquement les services décrits avec WSDL . SAWSDL ne spécifié pas un langage pour représenter les modèles sémantiques, mais plutôt il fournit un mécanisme à travers lequel des concepts appartenant à des modèles sémantiques existants (i.e., une ontologie de domaine) peuvent être référés à partir d'un document WSDL. SAWSDL propose des extensions à WSDL similaires à celles proposées par WSDL-S [3]. Sa particularité réside dans l'annotation supplémentaire des schémas XML.

2. http ://www.w3.org/TR/2003/WD-owl-features-20030331/

Plusieurs approches ont été proposées pour adresser le problème de la sémantisation des services REST. L'approche SA-REST [109] permet de sémantiser un service REST en ajoutant des annotations sémantique (RDFa) sur des ressources web qui décrivent la documentation de celui-ci (e.g., HTML, XHTML, etc). SAWADL [76,77] est une approche d'annotation pour les services REST. Elle propose des extensions pour la description WADL [49]. SOOWL-S (Social-oriented OWL-S) [80] est une extension de l'ontologie de services OWL-S. Elle est conçue dans le but de sémantiser les différents types d'APIs (e.g. SOAP, REST, JS, RSS,..) utilisées dans la construction des applications Mashup.

Une autre façon pour sémantiser les services web est d'utiliser les vues sémantiques. Cette catégorie d'approches est destinée beaucoup plus aux services orientés données (i.e., services DaaS). Dans [9], la sémantique d'un service web est décrite de façon déclarative en utilisant des vues RDF (vues paramétrées). Une vue RDF est une requête SPARQL définie à partir d'une ontologie de domaine. L'avantage de cette approche est qu'elle permet de définir non seulement la sémantique des entrées/sorites, mais aussi les relations sémantiques qui existent entre eux. Les vues RDF sont associées aux services en utilisant une des approches d'annotation, e.g., SAWSDL, WSDL-S, etc.

Les auteurs dans [120, 125] étendent le modèle de sémantisation **IOPE** (i.e., *Service Profile* de OWL-S) tout en utilisant les vues sémantiques. Dans cette approche, chaque paramètre d'entrée/sortie d'un service est annoté par une vue RDF, plutôt que par un simple concept ontologique qui décrit de façon globale la sémantique de celui-ci. Dans cette situation, les vues RDF jouent le rôle d'un mapping (i.e., une correspondance détaillée) entre l'ontologie de domaine et les entrées/sorties des services, ce qui permet d'assurer une sélection plus pertinente.

Matching des services web

Le matching des services permet d'identifier les services qui sont pertinents à la requête utilisateur. L'ensemble des services pertinents est obtenu en calculant des valeurs de similarité indiquant le degré de rapprochement entre les services et la requête. Le calcul de similarité se base soit sur les aspects syntaxiques des services soit sur les aspects sémantiques.

iMatcher1 [60] offre un système de matching syntaxique basé sur les profils de services (i.e., *Service Profile* de l'ontologie OWL-S). L'ensemble de profils de service est

stocké sous forme de graphes RDF sérialisé dans une base de données RDF avec une extension de RDQL, appelée iRDQL [13]. Le degré de similarité est calculé à partir de quatre métriques de similarité syntaxique : TFIDF (Term Frequency-Inverse Document Frequency), la distance de similarité de Levenshtein , la mesure du vecteur Cosinus et la mesure de divergence de Jensen-Shannon. Les résultats sont classés en fonction des scores numériques de ces mesures de similarités syntaxiques et d'un seuil défini par l'utilisateur.

Klusch et al. [61] proposent une approche de matching pour les services sémantique de type OWL-S. Dans cette approche, la requête utilisateur est définie à travers OWL-S de la même façon que les services. La comparaison entre la requête et les services se base uniquement sur la partie *Service Profile* de l'ontologie OWL-S (i.e., les propriétés IOPE). Les degrés de similarité sont calculés en se basant sur une approche hybride (OWL-MX) impliquant des techniques de similarité syntaxique (une approche à base d'IR) et sémantique (une approche à base de règles logiques). Dans un autre travail, Klusch et al. [63] adoptent l'approche OWL-MX pour l'ontologie de services WSMO.

SAWSDL-MX [62] est un matchmaker inspiré par les matchmakers OWLS-MX [61] et WSMO-MX [63]. Il est destiné pour les services qui sont sémantisés par le modèle SAWSDL. Il utilise à la fois une approche de matching logique basée sur le raisonnement par subsomption, et une approche de matching syntaxique basée sur les techniques de recherche d'information. Le processus de matching recouvre les éléments de description suivants : interface, operation, input et output. Par exemple, pour réaliser le matching des interfaces, le matchmaker effectue des matching sur des graphes bipartis dont les noeuds représentant des opérations du service. Les degrés de similarité dans une découverte de services avec SAWSDL-MX [62] peuvent être distingués en deux catégories : (1) Les degrés de similarité calculés par l'approche logique : Exact, Plug-in, Subsumes et Subsumed-by, et (2) Les degrés de similarité calculés par l'approche hybride :Subsumed-by (s'il a besoin de calcul de similarité syntaxique additionnelle) et Nearest-neighbour (pour compenser les appariements faux négatifs).

Les auteurs dans [103] proposent une approche de découverte qui se base sur les techniques de traitement du langage naturel (NLP). Dans cette approche, les services sont sémantiquement décrits par le modèle WSMO. Les requêtes utilisateurs sont spécifiées en langage naturel sous forme de mots clés. Le matching entre les mots clés de la requête

et les concepts ontologiques des services est réalisé en se basant sur plusieurs techniques, e.g., *part-of-speech tagging*, *lemmatization*, et *word sense disambiguation*.

3.2 Composition automatique de services web

La composition automatique permet un développement plus rapide des applications à base de services. Elle consiste à générer de façon transparente l'ensemble des compositions répondant à la requête utilisateur. Dans la suite nous classifions les approches de composition automatique en trois catégories : les approches à base de planification, les approches à base de méthodes formelles, et les approches basées sur la réécriture.

Approches à base de planification

La planification est l'un des domaines de l'Intelligence Artificielle qui permet de choisir et d'organiser des actions, en fonction d'un but donné. Elle produit un plan qui est présenté sous la forme d'une collection de descriptions d'opérateurs. Le mot planification est également utilisé dans plusieurs autres domaines, tels que l'économie, la politique, l'architecture et encore dans la vie de tous les jours.

Les auteurs dans [79] proposent une extension de Golog qui est un langage de programmation logique basé sur le calcul situationnel. L'idée présentée se base sur un ensemble de procédures génériques de haut niveau et sur la personnalisation des contraintes associées. Les procédures génériques représentent les besoins des utilisateurs. Les contraintes sont définies avec un langage de premier ordre. Les services sont modélisés comme des actions. Deux types d'actions sont possibles : des actions atomiques et des actions complexes. Ces dernières sont des compositions d'actions individuelles. Les auteurs proposent une approche qui permet à l'utilisateur de traduire les descriptions OWL-S des services impliqués en Prolog. Après cette traduction, le système utilise des règles d'inférences logiques pour automatiser le choix d'un service pour chaque action en respectant les contraintes associées.

Medjahed et al. [81] présentent une approche de planification à base de règles. Les services composites sont générés à partir des déclarations de haut niveau. Ils décrivent une approche de composition en quatre phases. Ils présentent des règles syntaxiques et des règles sémantiques. Ces règles décrivent les attributs nécessaires à la composition des

services impliqués.

Shankar et al. [94]proposent SWORD, un outil de composition basé sur le *rule-based planning*. SWORD raisonne sur des services décrits selon le modèle Entity-Relationship (ER). Il associe aux services des règles logiques de type *Horn*. La création d'un service web composite est ramenée à la spécification d'un état de départ et d'un état d'arrivée. Les états intermédiaires sont générés par le moteur de planification à partir des descriptions des services disponibles.

Le *Planning Domain Definition Language* (*PDDL*) est un langage de planification largement adopté par les communautés de l'Intelligence Artificielle. Dans [122], McDermott propose une approche de composition à base de *PDDL*qui est étendu pour formaliser les entrées et les sorties des services. Il propose la notion de *value of an action* qui modélise l'exécution d'un service. Elle exprime des changements qui peuvent avoir lieu dans l'environnement. Dans cette approche, la description DAML-S (ancienne version de OWL-S) est traduite en *PDDL* afin de modéliser le problème de composition de services comme étant un problème de satisfaction d'un planning. Les plans sont générés à l'aide de la technique de planification *Estimated-regression*.

Un travail similaire est introduit dans [14] dans le but de composer les services OWL-S en se basant sur *PDDL*. Ce travail fournit une solution sur la façon de traduire le process model de OWL-S en des actions *PDDL*.

Approches à base de méthodes formelles

Les méthodes formelles à savoir : les automates, les réseaux Petri, etc ont été largement utilisées dans la composition des services.

Les automates ou systèmes de transition d'états annotés offrent une sémantique précise et adaptée à la représentation de la composition. Les techniques de raisonnement applicables aux automates permettent de vérifier la structure de la composition et sa cohérence. Un automate est constitué d'un ensemble d'états, d'un ensemble de transitions annotées entre états, et d'un ensemble d'actions. Les annotations des transitions contiennent des actions représentant le passage d'un état à un autre.

Bordeaux et al. [16] suggèrent de représenter le comportement du service par un système de transition d'états. Cela revient à modéliser un service par un ensemble d'états

reliés par des actions. Les auteurs précisent que ce modèle est déterministe dans la mesure ou "une action qui a lieu à un état donné conduit à un état déterminé".

Un réseau Petri est un ensemble de *places* et de *transitions* reliées par des arcs. L'aspect dynamique du réseau Petri est décrit par un ensemble de règles. Chaque règle définit (i) les conditions à valider pour qu'une transaction ait lieu, et (ii) les effets de cette transaction. Les réseaux Petri (Petri Nets) fournissent une représentation précise et sémantiquement riche de l'ordre d'exécution des activités. Ils permettent un raisonnement plus efficace sur une composition donnée.

Narayanan et al. [83] proposent un outil qui génère automatiquement un réseau de Petri pour les compositions qui sont exprimée par le *Service Model* de OWL-S. Cet outil vérifie les propriétés de la composition générée. Il juge la consistance du graphe modélisant le flux d'interaction. Il détermine, par exemple, si tous les états (nœuds) peuvent être atteints, si les contraintes décrites sont respectées ou si des impasses existent, etc.

Ouyang et al. [90] proposent une traduction des contrôles de flux BPEL en réseaux de Petri annotés. Cette traduction permet de raisonner automatiquement sur la structure du graphe d'interaction et sur ses propriétés.

Approches à base de réécriture

Dans les systèmes d'intégration de données classiques, la réécriture de requête est le processus qui permet de reformuler une requête globale (i.e., définie à partir du schéma médiateur) en termes de vues. Chaque vue décrit les correspondances (i.e., mapping) entre le schéma médiateur et le schéma d'une source de données locale. Les techniques de réécriture de requêtes ont été largement étudiées par la communauté des bases de données. Un état de l'art sur les principaux algorithmes de réécriture est disponible dans [50].

Dans le contexte des services web, plusieurs approches [8, 9, 22, 70, 82, 117, 124] ont adopté les techniques de réécriture de requêtes pour la composition des services. Dans la plupart des solutions proposées, la réécriture de requêtes se base sur la sémantique associée aux services.

Thakkar et al. [117] proposent une approche de composition de services à base de médiateur. Les requêtes utilisateur sont formulées à partir d'un ensemble de prédicats de domaine (i.e., l'équivalant du schéma médiateur) définies par un expert. Cette approche

utilise le principe des systèmes d'intégration de données *LAV* (i.e., Local As View), où chaque service est représenté par une vue décrite à partir des prédicats de domaine tout en utilisant la notation Datalog. La composition des services est réalisée de façon automatique en se basant sur l'algorithme de réécriture *Inverse Rules* [35]. Les auteurs proposent aussi un algorithme appelé *tuple-level filtering*. Cet algorithme implémente une approche à base de contrainte dans le but d'optimiser le plan d'exécution des compositions générées.

Une autre approche de composition de services à base de réécriture est proposée dans [8]. Dans cette approche, les services sont sémantiquement représentés par des vues conjonctives décrites dans le langage *CARIN* [67]. Ce langage est une extension de *DL* (Description Logics) destiné à la description des sources de données. Les requêtes utilisateurs sont représentées dans le langage *CARIN* à partir d'une ontologie médiatrice. Les requêtes sont réécrites à travers les vues conjonctives des services en utilisant un algorithme de réécriture sémantiques à base de DL [46].

Les auteurs dans [9], étudient le problème de la réécriture de requêtes à travers des services web qui sont sémantiquement décrits par des vues RDF (i.e., requête SPARQL). L'algorithme de réécriture proposé est divisé en deux parties. (*i*) La première phase consiste à identifier les services pertinents. Un service est pertinent, si dans la vue RDF associée, il existe au moins un ClassNode (resp, ObjectProperty) qui match avec un autre ClassNode (resp, ObjectProperty) dans la requête utilisateur. Les ClassNodes et les ObjectProperties constituent l'ensemble des sous-buts d'une requête SPARQL. (*ii*) La deuxième phase permet de générer l'ensemble des compositions candidates. Cet ensemble est obtenu en créant toutes les combinaisons possibles entre les services pertinents, et en éliminant celles dont l'union des vues RDF des services impliqués ne couvre pas de façon complète la requête utilisateur.

WenFeng et al. [124] proposent une approche de réécriture pour les services web OWL-S. Ils se basent uniquement sur partie IOPE (i.e., service profile) de l'ontologie OWL-S. la réécriture de requête est réalisée en proposant une extension pour l'algorithme MiniCon [95]. Les auteurs proposent aussi un ensemble de règles qui permettent de transformer un plan de composition générique en un script BPEL.

Les autres dans [82], combinent les techniques de réécriture de requêtes et de configurations dans le but de faciliter la composition automatique des services web sémantiques.

Les auteurs dans [22], utilisent les techniques de réécriture de requêtes pour assurer le raffinement automatique des compositions de services. L'approche proposée se base aussi bien sur les aspects sémantiques des services que sur leurs propriétés fonctionnelles et non-fonctionnelles.

3.3 Discussion

Plusieurs modèles ont été proposés pour la sémantisation des services web. Ces modèles peuvent être distingués selon la nature des services. Par exemple, les modèles OWL-S [7], WSMO [100], SAWSDL [65], SAWADL, SAWADL [77], etc sont appropriés pour les services orientés traitement, i.e., les capacités d'un service sont représentées par ses entrés sorties. En revanche, ces modèles ne peuvent pas être utilisés pour décrire la sémantique des services orientés données (services DaaS). Ceci s'explique par le fait que la sémantique d'un service DaaS ne dépend pas seulement de ses entrés, mais aussi de la relation sémantique qui existe eux. Les modèles à base de vue sémantique [9, 120, 125] décrivent la sémantique d'un service DaaS de façon déclarative tout en prenant en considération les relations entre ses entrés sorties. Autrement dit, une vue sémantique décrit la façon (i.e., la requête) dont un service accède à une source de données.

Dans notre travail, les services sont sémantiquement décrits par des vues sémantiques. Cependant, et à la différence des travaux existant, nous traitons le problème d'incertitude lié à l'interprétation des services. Dans une telle situation, chaque service est décrit par plus d'une vue sémantique, chacune avec une probabilité. L'ensemble des vues sémantiques associées à un service DaaS représente les différentes interprétations possibles des données encapsulées par celui-ci.

Les approches de composition des services se différent selon plusieurs caractéristiques, telles que la nature des services (i.e., orientés traitement, orientés données), la description des services (e.g., OWL-S, WSMO, Vue sémantique, etc), la méthode de combinaison (e.g., planification, réseau Petri, réécriture, etc), et la structure de la requête (e.g., requête service, requête déclarative, etc). Cependant, dans la littérature, il n'y a aucun travail qui traite la sémantique incertaine des services. Dans le cadre de cette thèse, nous proposons une approche pour la représentation et la composition des services DaaS à sémantique incertaine. Les services sont représentés par un registre de services probabiliste qui décrit

leurs vues sémantiques probabilistes. Pour la composition des services incertains, nous étendons une approche à base de réécriture [9], dans laquelle les requêtes utilisateurs et les sémantiques des services ont une forme déclarative. En outre, nous introduisons le concept *Interprétation de Composition* qui permet de calculer la probabilité des compositions, et ainsi vérifier sa validité, i.e., est-ce qu'une composition satisfait les corrélations qui existent entre les vues ?.

Chapitre 7

Conclusion Générale

1 Résumé des contributions

Ces dernières années ont vu un intérêt croissant pour l'utilisation des services web DaaS comme un support fiable pour l'édition et le partage des données entre les organisations. Dans ce contexte, les requêtes utilisateurs sont définies de façon déclarative, et résolues par la composition des services DaaS. Le processus de composition peut être automatisé en se basant sur la sémantique des services. La sémantique d'un service décrit l'interprétation des données encapsulées par celui-ci. Dans la littérature, le modèle déclaratif à base de vues sémantiques (i.e., *Vue RDF*) est le plus approprié pour décrire l'interprétation des services DaaS. Cependant, dans certain cas, la sémantique d'un service est incertaine. Cela est dû à plusieurs raisons, e.g., des services fournis sans documentation, l'utilisation des techniques de matching approximatives, etc.

Dans cette thèse, nous proposons un modèle probabiliste permettant la représentation et la composition des services DaaS dont l'interprétation des données encapsulées est incertaine. Un service incertain est sémantiquement décrit par plusieurs vues sémantiques, chacune avec une probabilité. Les vues sémantiques possibles d'un service ainsi que leurs probabilités peuvent être générées en se basant sur différentes techniques de similarité (e.g., matching à base de nom, matching à base de contenu, etc) qui permettent de calculer le degré de rapprochement entre les entrées/sorties d'un service et les éléments d'une ontologie de domaine.

Les services ainsi que leurs vues sémantiques possibles sont représentés dans une table probabiliste <*service*, *view*, *prob*> appelée *Registre de Services Probabiliste* (PSR). Formellement, les corrélations dans PSR peuvent être représentées par deux modèles différents : le modèle *BID*, et le modèle *Bayésien*. Le modèle *BID* ne permet de décrire qu'un seul type de corrélation (i.e., disjonction) entre l'ensemble des vues sémantiques, et cela lorsque les services correspondants n'accèdent pas aux mêmes sources de données. Cependant, lorsque les services DaaS partagent entre les mêmes sources de données, plusieurs corrélations complexes sont possibles. Dans ce cas, PSR est décrit par un réseau bayésien qui est le seul modèle capable de fournir une représentation générique de ces corrélations. Un registre de services probabiliste PSR est interprété comme étant une distribution de probabilité sur un ensemble de registres de services possibles. Le nombre, la structure, et la probabilité des registres de services possibles sont définies en fonction des corrélations existantes (i.e., *BID*, réseau bayésien).

Evaluer une requête utilisateur Q à travers un registre de services probabiliste revient à l'évaluer sur chaque registre de services possible. L'évaluation d'une requête peut accepter deux interprétations : la *Sémantique des Résultats Possibles* où les compositions possibles dans chaque monde possible sont retournées (associées avec la probabilité du monde), et la *Sémantique des Compositions possibles*, où les compositions possibles dans tous les mondes sont retournées avec leurs probabilités (agrégées). Ces sémantiques s'appliquent de la même façon pour les deux modèles de PSR (i.e., *BID*, réseau bayésien).

L'évaluation des requêtes à travers l'ensemble des registres de services possibles est clairement intractable. Ceci s'explique par le fait que le nombre de ces registres possibles est exponentiel. Dans ce sens, nous avons proposé deux algorithmes (un pour PSR avec BID, et un autre pour PSR bayésien) permettant d'évaluer efficacement les requêtes utilisateurs (i.e., dans un temps raisonnable), et cela sans avoir besoin de matérialiser l'ensemble des registres de services possibles. De plus, le résultat obtenu (i.e., compositions et leurs probabilités) par notre solution est en adéquation avec la *Sémantique des Compositions possibles*. Nous proposons aussi un algorithme qui implémente une stratégie qui assure la génération efficace de l'ensemble des Top-k compositions.

Nous avons implémenté deux systèmes de composition de services incertains (i.e., à sémantique non-corrélée, et à sémantique corrélée). Nous avons mené une série d'expérimentation

pour évaluer la performance de nos différents algorithmes de composition. Les résultats obtenus montrent l'efficacité et la scalabilité de nos solutions proposées.

2 Perspectives

A la suite de cette thèse, nous dégageons un ensemble de perspectives futures répondant à trois objectifs principaux : (*i*) La réduction de l'incertitude des services ; (*ii*) la composition des services dont les données retournées sont incertaines ; (*iii*) et la matérialisation des compositions incertaines. Dans ce qui suit nous détaillons chacune de ces perspectives.

Réduction de l'incertitude des services. Une première perspective consiste à utiliser le concept *Crowdsourcing* dans le but de réduire (voire éliminer) l'incertitude des services. Le *Crowdsourcing* est un des domaines émergents de la gestion des connaissances, qui consiste en l'utilisation de la créativité, de l'intelligence et du savoir-faire d'un grand nombre de personnes, en sous-traitance, pour réaliser certaines tâches traditionnellement effectuées par des experts.

Composition des services dont les données retournées sont incertaines. Dans cette thèse, nous traitons l'incertitude liée à la sémantique des services, les données retournées par ces derniers sont supposées certaines. Comme un futur travail, nous comptons étendre notre approche pour gérer l'incertitude au niveau des données, i.e., comment composer des services retournant des données incertaines (e.g., données probabilistes).

Matérialisation des compositions incertaines. Dans le contexte des services DaaS, une composition résultante peut être matérialisée en tant qu'un nouveau service (i.e., service composite), et cela dans le but d'optimiser le processus de réécriture et d'exécution. Les vues sémantiques possibles de ce nouveau service correspondent aux interprétations possibles de la composition en question. Cependant, il est nécessaire de calculer le *lineage* de ces vues sémantiques, et cela afin d'assurer la cohérence et la non-ambigüité des compositions impliquant des services composites. Le *lineage* d'une vue sémantique possible d'un service composite est une formule propositionnelle qui décrit ses provenances, i.e., les vues sémantiques qui la construisent. Dans un futur travail, nous nous intéresserons au calcul de ce *lineage*, ainsi qu'à la composition des services dont la sémantique est définie par une

formule propositionnelle.

Bibliographie

[1] S. Abiteboul, P. C. Kanellakis, and G. Grahne. On the representation and querying of sets of possible worlds. In *SIGMOD Conference*, pages 34–48, 1987.

[2] S. Agrawal, S. Chaudhuri, and V. R. Narasayya. Automated selection of materialized views and indexes in sql databases. In *Proceedings of the 26th International Conference on Very Large Data Bases*, VLDB '00, pages 496–505, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc.

[3] R. Akkiraju, J. Farrell, J. A. Miller, M. Nagarajan, A. Sheth, K. Verma, R. Akkiraju, J. Farrell, J. A. M. Iller, M. Nagarajan, A. Sheth, and K. Verma. Web service semantics – wsdl-s. In *In W3C Workshop on Frameworks for Semantic in Web Services*, 2005.

[4] R. Alarcón and E. Wilde. Restler : Crawling restful services. In *Proceedings of the 19th International Conference on World Wide Web*, WWW '10, pages 1051–1052, New York, NY, USA, 2010. ACM.

[5] L. Antova, C. Koch, and D. Olteanu. 10^{10^6} worlds and beyond : Efficient representation and processing of incomplete information. In *ICDE*, pages 606–615, 2007.

[6] L. Antova, C. Koch, and D. Olteanu. Maybms : Managing incomplete information with probabilistic world-set decompositions. In *Data Engineering, 2007. ICDE 2007. IEEE 23rd International Conference on*, pages 1479–1480, April 2007.

[7] M. A. Aslam, S. Auer, J. Shen, and M. Herrmann. Expressing business process models as owl-s ontologies. In *Proceedings of the 2006 International Conference on Business Process Management Workshops*, BPM'06, pages 400–415, Berlin, Heidelberg, 2006. Springer-Verlag.

[8] S. Bao, L. Zhang, C. Lin, and Y. Yu. A semantic rewriting approach to automatic information providing web service composition. In *ASWC*, pages 488–500, 2006.

[9] M. Barhamgi, D. Benslimane, and B. Medjahed. A query rewriting approach for web service composition. *IEEE T. Services Computing*, 3(3) :206–222, 2010.

[10] D. K. Barry and P. J. Gannon. *Web Services and Service-Oriented Architecture : The Savvy Manager's Guide*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2003.

[11] O. Benjelloun, A. D. Sarma, A. Y. Halevy, M. Theobald, and J. Widom. Databases with uncertainty and lineage. *VLDB J.*, 17(2) :243–264, 2008.

[12] T. Bernecker, T. Emrich, H.-P. Kriegel, M. Renz, and A. Züfle. Probabilistic ranking in fuzzy object databases. In *CIKM*, pages 2647–2650, 2012.

[13] A. Bernstein and C. Kiefer. Imprecise rdql : Towards generic retrieval in ontologies using similarity joins. In *Proceedings of the 2006 ACM Symposium on Applied Computing*, SAC '06, pages 1684–1689, New York, NY, USA, 2006. ACM.

[14] Y. Bo and Q. Zheng. A method of semantic web service composition based pddl. In *Service-Oriented Computing and Applications (SOCA), 2009 IEEE International Conference on*, pages 1–4, Jan 2009.

[15] C. Böhm, M. Freitag, and A. Heise. Integrating open government data for transparency. In *WWW*, pages 21–24, 2012.

[16] L. Bordeaux, G. Salaün, D. Berardi, and M. Mecella. When are two web services compatible ? In M.-C. Shan, U. Dayal, and M. Hsu, editors, *Technologies for E-Services*, volume 3324 of *Lecture Notes in Computer Science*, pages 15–28. Springer Berlin Heidelberg, 2005.

[17] P. Bosc and O. Pivert. About projection-selection-join queries addressed to possibilistic relational databases. *IEEE T. Fuzzy Systems*, 13(1) :124–139, 2005.

[18] M. J. Carey, N. Onose, and M. Petropoulos. Data services. *Commun. ACM*, 55(6) :86–97, 2012.

[19] R. Cheng, J. Gong, D. W. Cheung, and J. Cheng. Evaluating probabilistic queries over uncertain matching. In *IEEE 28th International Conference on Data Enginee-*

ring (ICDE 2012), Washington, DC, USA (Arlington, Virginia), 1-5 April, 2012, pages 1096–1107, 2012.

[20] L. Chiticariu, P. G. Kolaitis, and L. Popa. Interactive generation of integrated schemas. In *SIGMOD Conference*, pages 833–846, 2008.

[21] S. Choenni, H. E. Blok, and E. Leertouwer. Handling uncertainty and ignorance in databases : A rule to combine dependent data. In *DASFAA*, pages 310–324, 2006.

[22] U. Costa, M. Ferrari, M. Musicante, and S. Robert. Automatic refinement of service compositions. In F. Daniel, P. Dolog, and Q. Li, editors, *Web Engineering*, volume 7977 of *Lecture Notes in Computer Science*, pages 400–407. Springer Berlin Heidelberg, 2013.

[23] R. G. Cowell, A. P. Dawid, S. L. Lauritzen, and D. J. Spiegelhalter. *Probabilistic Networks and Expert Systems : Exact Computational Methods for Bayesian Networks.* Springer Publishing Company, Incorporated, 1st edition, 2007.

[24] N. N. Dalvi, C. Re, and D. Suciu. Queries and materialized views on probabilistic databases. *J. Comput. Syst. Sci.*, 77(3) :473–490, 2011.

[25] N. N. Dalvi and D. Suciu. Efficient query evaluation on probabilistic databases. In *VLDB*, pages 864–875, 2004.

[26] N. N. Dalvi and D. Suciu. Efficient query evaluation on probabilistic databases. *VLDB J.*, 16(4) :523–544, 2007.

[27] A. Das Sarma, X. Dong, and A. Halevy. Bootstrapping pay-as-you-go data integration systems. In *SIGMOD*, pages 861–874, New York, NY, USA, 2008. ACM.

[28] R. Dechter. Bucket elimination : A unifying framework for probabilistic inference. In *UAI*, pages 211–219, 1996.

[29] A. Deshpande, C. Guestrin, S. R. Madden, J. M. Hellerstein, and W. Hong. Model-driven data acquisition in sensor networks. In *Proceedings of the Thirtieth International Conference on Very Large Data Bases - Volume 30*, VLDB '04, pages 588–599. VLDB Endowment, 2004.

[30] L. Ding, T. Lebo, J. S. Erickson, D. DiFranzo, G. T. Williams, X. Li, J. Michaelis, A. Graves, J. Zheng, Z. Shangguan, J. Flores, D. L. McGuinness, and J. A. Hendler.

Twc logd : A portal for linked open government data ecosystems. *J. Web Sem.*, 9(3) :325–333, 2011.

[31] H.-H. Do and E. Rahm. Coma : A system for flexible combination of schema matching approaches. In *VLDB*, pages 610–621. VLDB Endowment, 2002.

[32] C. Domshlak, A. Gal, and H. Roitman. Rank aggregation for automatic schema matching. *IEEE Trans. on Knowl. and Data Eng.*, 19(4) :538–553, Apr. 2007.

[33] X. Dong, A. Y. Halevy, and C. Yu. Data integration with uncertainty. In *Proceedings of the 33rd International Conference on Very Large Data Bases*, VLDB '07, pages 687–698. VLDB Endowment, 2007.

[34] X. L. Dong, A. Halevy, and C. Yu. Data integration with uncertainty. *The VLDB Journal*, 18(2) :469–500, Apr. 2009.

[35] O. M. Duschka and M. R. Genesereth. Query planning in infomaster. In *Proceedings of the 1997 ACM Symposium on Applied Computing*, SAC '97, pages 109–111, New York, NY, USA, 1997. ACM.

[36] M. Dylla, I. Miliaraki, and M. Theobald. Top-k query processing in probabilistic databases with non-materialized views. In *ICDE*, pages 122–133, 2013.

[37] W. Emmerich and N. Kaveh. Component technologies : Java beans, com, corba, rmi, ejb and the corba component model. In *Proceedings of the 8th European Software Engineering Conference Held Jointly with 9th ACM SIGSOFT International Symposium on Foundations of Software Engineering*, ESEC/FSE-9, pages 311–312, New York, NY, USA, 2001. ACM.

[38] R. Fagin, A. Lotem, and M. Naor. Optimal aggregation algorithms for middleware. In *Proceedings of the Twentieth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, PODS '01, pages 102–113, New York, NY, USA, 2001. ACM.

[39] D. Fensel, C. Bussler, Y. Ding, and B. Omelayenko. The web service modeling framework wsmf. *Electronic Commerce Research and Applications*, pages 113–137, 2002.

[40] R. T. Fielding. *Architectural Styles and the Design of Network-based Software Architectures*. PhD thesis, UNIVERSITY OF CALIFORNIA,IRVINE, 2000.

[41] R. Fink, D. Olteanu, and S. Rath. Providing support for full relational algebra in probabilistic databases. In *ICDE*, pages 315–326, 2011.

[42] N. Fuhr and T. Rölleke. A probabilistic relational algebra for the integration of information retrieval and database systems. *ACM.Inf.Syst.*, pages 32–66, 1997.

[43] A. Gal. Interpreting similarity measures : Bridging the gap between schema matching and data integration. In *Data Engineering Workshop, 2008. ICDEW 2008. IEEE 24th International Conference on*, pages 278–285, April 2008.

[44] A. Gal, A. Anaby-Tavor, A. Trombetta, and D. Montesi. A framework for modeling and evaluating automatic semantic reconciliation. *The VLDB Journal*, 14(1) :50–67, Mar. 2005.

[45] A. Gal, M. V. Martinez, G. I. Simari, and V. S. Subrahmanian. Aggregate query answering under uncertain schema mappings. In *Proceedings of the 25th International Conference on Data Engineering, ICDE 2009, Shanghai, China*, pages 940–951, 2009.

[46] F. Goasdoué and M.-C. Rousset. Answering queries using views : A krdb perspective for the semantic web. *ACM Trans. Internet Technol.*, 4(3) :255–288, Aug. 2004.

[47] T. J. Green and V. Tannen. Models for incomplete and probabilistic information. *IEEE Data Eng. Bull.*, 29(1) :17–24, 2006.

[48] R. Gupta and S. Sarawagi. Creating probabilistic databases from information extraction models. In *VLDB*, pages 965–976, 2006.

[49] M. Hadley. Web application description language. Technical report, W3C, Sun Microsystems, Inc., 4150 Network Circle, Santa Clara, California 95054, U.S.A, 2009.

[50] A. Y. Halevy. Answering queries using views : A survey. *The VLDB Journal*, 10(4) :270–294, Dec. 2001.

[51] B. He and K. C.-C. Chang. Statistical schema matching across web query interfaces. In *Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data*, SIGMOD '03, pages 217–228, New York, NY, USA, 2003. ACM.

[52] B. He and K. C.-C. Chang. Automatic complex schema matching across web query interfaces : A correlation mining approach. *ACM Trans. Database Syst.*, 31(1) :346–395, 2006.

[53] B. He, K. C.-C. Chang, and J. Han. Discovering complex matchings across web query interfaces : A correlation mining approach. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '04, pages 148–157, New York, NY, USA, 2004. ACM.

[54] A. Heise and al. Integrating open government data with stratosphere for more transparency. *J. Web Sem.*, 14 :45–56, 2012.

[55] C. Huang and A. Darwiche. Inference in belief networks : A procedural guide. *International Journal of Approximate Reasoning*, 15 :225–263, 1996.

[56] R. Jampani, F. Xu, M. Wu, L. L. Perez, C. M. Jermaine, and P. J. Haas. Mcdb : a monte carlo approach to managing uncertain data. In *SIGMOD Conference*, pages 687–700, 2008.

[57] S. R. Jeffery, M. N. Garofalakis, and M. J. Franklin. Adaptive cleaning for rfid data streams. In *VLDB*, pages 163–174, 2006.

[58] A. Jha and D. Suciu. Probabilistic databases with markoviews. *Proc. VLDB Endow.*, 5(11) :1160–1171, July 2012.

[59] B. Kanagal and A. Deshpande. Lineage processing over correlated probabilistic databases. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data*, SIGMOD '10, pages 675–686, New York, NY, USA, 2010. ACM.

[60] C. Kiefer and A. Bernstein. The creation and evaluation of isparql strategies for matchmaking. In *The Semantic Web : Research and Applications, 5th European Semantic Web Conference, ESWC 2008, Tenerife, Canary Islands, Spain, June 1-5, 2008, Proceedings*, pages 463–477, 2008.

[61] M. Klusch, B. Fries, and K. Sycara. Automated semantic web service discovery with owls-mx. In *Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multiagent Systems*, AAMAS '06, pages 915–922, New York, NY, USA, 2006. ACM.

[62] M. Klusch, P. Kapahnke, and I. Zinnikus. Hybrid adaptive web service selection with SAWSDL-MX and wsdl-analyzer. In *The Semantic Web : Research and Applications,*

6th European Semantic Web Conference, ESWC 2009, Heraklion, Crete, Greece, May 31-June 4, 2009, Proceedings, pages 550–564, 2009.

[63] M. Klusch and F. Kaufer. Wsmo-mx : A hybrid semantic web service matchmaker. *Web Intelli. and Agent Sys.*, 7(1) :23–42, Jan. 2009.

[64] J. Kopecký, K. Gomadam, and T. Vitvar. hrests : An html microformat for describing restful web services. In *Proceedings of the 2008 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology - Volume 01*, WI-IAT '08, pages 619–625, Washington, DC, USA, 2008. IEEE Computer Society.

[65] J. Kopecký, T. Vitvar, C. Bournez, and J. Farrell. Sawsdl : Semantic annotations for wsdl and xml schema. *IEEE Internet Computing*, 11(6) :60–67, 2007.

[66] J. D. Lafferty, A. McCallum, and F. C. N. Pereira. Conditional random fields : Probabilistic models for segmenting and labeling sequence data. In *ICML*, pages 282–289, 2001.

[67] A. Y. Levy and M.-C. Rousset. Combining horn rules and description logics in {CARIN}. *Artificial Intelligence*, 104(1–2) :165 – 209, 1998.

[68] J. Li and A. Deshpande. Consensus answers for queries over probabilistic databases. In *Proceedings of the Twenty-eighth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, PODS '09, pages 259–268, New York, NY, USA, 2009. ACM.

[69] Q. A. Liang and H. Lam. Web service matching by ontology instance categorization. In *IEEE SCC (1)*, pages 202–209, 2008.

[70] J. Lu, Y. Yu, and J. Mylopoulos. A lightweight approach to semantic web service synthesis. In *Web Information Retrieval and Integration, 2005. WIRI '05. Proceedings. International Workshop on Challenges in*, pages 240–247, April 2005.

[71] A. L. Machado and J. M. Oliveira. Digo :an opendata architecture for e-government. In *EDOCW*, pages 448–456, 2011.

[72] M. Magnani and D. Montesi. Preference-based uncertain data integration. In *Proceedings of the 16th International Conference on Knowledge Engineering : Practice and Patterns*, EKAW '08, pages 136–145, Berlin, Heidelberg, 2008. Springer-Verlag.

[73] M. Magnani, N. Rizopoulos, P. McBrien, and D. Montesi. Schema integration based on uncertain semantic mappings. In *Conceptual Modeling - ER 2005, 24th International Conference on Conceptual Modeling, Klagenfurt, Austria, October 24-28, 2005, Proceedings*, pages 31–46, 2005.

[74] A. Malki, M. Barhamgi, S. M. Benslimane, D. Benslimane, and M. Malki. Composing Data Services with Uncertain Semantics. *IEEE Transactions on Knowledge and Data Engineering*, à paraître.

[75] A. Malki, D. Benslimane, S. M. Benslimane, M. Barhamgi, M. Malki, P. Ghodous, and K. Drira. Data Services with uncertain and correlated semantics. *WORLD WIDE WEB Journal*, à paraître.

[76] A. Malki and S. M. Benslimane. Building semantic mashup. In *ICWIT*, pages 40–49, 2012.

[77] A. Malki and S. M. Benslimane. Semantic cloud : Building dynamic mashup in cloud environment. *IJITWE*, 8(4) :20–35, 2013.

[78] R. Mateescu and R. Dechter. And/or cutset conditioning. IJCAI'05, pages 230–235, USA, 2005. Morgan Kaufmann Publishers Inc.

[79] S. A. McIlraith and T. C. Son. Adapting golog for composition of semantic web services. In *Proceedings of the Eights International Conference on Principles and Knowledge Representation and Reasoning (KR-02), Toulouse, France, April 22-25, 2002*, pages 482–496, 2002.

[80] G. Meditskos and N. Bassiliades. A combinatory framework of web 2.0 mashup tools, owl-s and uddi. *Expert Syst. Appl.*, 38(6) :6657–6668, June 2011.

[81] B. Medjahed, A. Bouguettaya, and A. K. Elmagarmid. Composing web services on the semantic web. *The VLDB Journal*, 12(4) :333–351, Nov. 2003.

[82] A. Mesmoudi, M. Mrissa, and M. Hacid. Combining configuration and query rewriting for web service composition. In *Web Services (ICWS), 2011 IEEE International Conference on*, pages 113–120, July 2011.

[83] S. Narayanan and S. A. McIlraith. Simulation, verification and automated composition of web services. In *Proceedings of the Eleventh International World Wide Web Conference, WWW 2002, May 7-11, 2002, Honolulu, Hawaii*, pages 77–88, 2002.

[84] F. Niu, C. Ré, A. Doan, and J. Shavlik. Tuffy : Scaling up statistical inference in markov logic networks using an rdbms. *Proc. VLDB Endow.*, 4(6) :373–384, Mar. 2011.

[85] H. Nottelmann, , H. Nottelmann, and U. Straccia. splmap : A probabilistic approach to schema matching. In *In 27 th European Conference on Information Retrieval (ECIR'05*, pages 81–95. Springer Verlag, 2005.

[86] H. Nottelmann and U. Straccia. Information retrieval and machine learning for probabilistic schema matching. *Inf. Process. Manage.*, 43(3) :552–576, May 2007.

[87] D. Olteanu, J. Huang, and C. Koch. Approximate confidence computation in probabilistic databases. In *ICDE*, pages 145–156, 2010.

[88] D. Olteanu, C. Koch, and L. Antova. World-set decompositions : Expressiveness and efficient algorithms. *Theor. Comput. Sci.*, 403(2-3) :265–284, Aug. 2008.

[89] J. O'Sullivan, D. Edmond, and A. Ter Hofstede. What's in a service ? *Distrib. Parallel Databases*, 12(2-3) :117–133, Sept. 2002.

[90] C. Ouyang, E. Verbeek, W. M. P. van der Aalst, S. Breutel, M. Dumas, and A. H. M. ter Hofstede. Formal semantics and analysis of control flow in ws-bpel. *Sci. Comput. Program.*, 67(2-3) :162–198, July 2007.

[91] M. P. Papazoglou. Service -oriented computing : Concepts, characteristics and directions. In *Proceedings of the Fourth International Conference on Web Information Systems Engineering*, WISE '03, pages 3–, Washington, DC, USA, 2003. IEEE Computer Society.

[92] C. Pautasso, O. Zimmermann, and F. Leymann. Restful web services vs. "big"' web services : Making the right architectural decision. In *Proceedings of the 17th International Conference on World Wide Web*, WWW '08, pages 805–814, New York, NY, USA, 2008. ACM.

[93] J. Pearl. *Probabilistic Reasoning in Intelligent Systems : Networks of Plausible Inference*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1988.

[94] S. R. Ponnekanti and A. Fox. SWORD : A Developer Toolkit for Web Service Composition. In *World Wide Web Conference Series*, 2002.

[95] R. Pottinger and A. Halevy. Minicon : A scalable algorithm for answering queries using views. *The VLDB Journal*, 10(2-3) :182–198, Sept. 2001.

[96] B. Qin and Y. Xia. Generating efficient safe query plans for probabilistic databases. *Data Knowledge Engineering*, 67(3) :485 – 503, 2008.

[97] C. Re, N. N. Dalvi, and D. Suciu. Efficient top-k query evaluation on probabilistic data. In *ICDE*, pages 886–895, 2007.

[98] C. Ré, J. Letchner, M. Balazinska, and D. Suciu. Event queries on correlated probabilistic streams. In *SIGMOD Conference*, pages 715–728, 2008.

[99] C. Ré and D. Suciu. Materialized views in probabilistic databases : For information exchange and query optimization. In *Proceedings of the 33rd International Conference on Very Large Data Bases*, VLDB '07, pages 51–62. VLDB Endowment, 2007.

[100] D. Roman, U. Keller, H. Lausen, J. de Bruijn, R. Lara, M. Stollberg, A. Polleres, C. Feier, C. Bussler, and D. Fensel. Web service modeling ontology. *Appl. Ontol.*, 1(1) :77–106, Jan. 2005.

[101] M. Sabesan and T. Risch. Adaptive parallelization of queries over dependent service calls. In *ICDE*, pages 1725–1732, 2009.

[102] L. Á. Sabucedo, L. E. Anido-Rifón, R. M. Pérez, and J. M. Santos-Gago. Providing standard-oriented data models and interfaces to egovernment services : A semantic-driven approach. *Computer Standards & Interfaces*, 31(5) :1014–1027, 2009.

[103] J. Sangers, F. Frasincar, F. Hogenboom, and V. Chepegin. Semantic web service discovery using natural language processing techniques. *Expert Systems with Applications*, 40(11) :4660 – 4671, 2013.

[104] A. D. Sarma, O. Benjelloun, A. Halevy, and J. Widom. Working models for uncertain data. In *Proceedings of the 22Nd International Conference on Data Engineering*, ICDE '06, pages 7–, Washington, DC, USA, 2006. IEEE Computer Society.

[105] A. D. Sarma, X. L. Dong, and A. Y. Halevy. Uncertainty in data integration and dataspace support platforms. In *Schema Matching and Mapping*, pages 75–108. 2011.

[106] A. Segev and Q. Z. Sheng. Bootstrapping ontologies for web services. *IEEE T. Services Computing*, 5(1) :33–44, 2012.

[107] P. Sen and A. Deshpande. Representing and querying correlated tuples in probabilistic databases. In *ICDE*, pages 596–605, 2007.

[108] P. Sen, A. Deshpande, and L. Getoor. Prdb : managing and exploiting rich correlations in probabilistic databases. *VLDB J.*, 18(5) :1065–1090, 2009.

[109] A. Sheth, K. Gomadam, and J. Lathem. Sa-rest : Semantically interoperable and easier-to-use services and mashups. *Internet Computing, IEEE*, 11(6) :91–94, Nov 2007.

[110] A. P. Sheth and A. Ranabahu. Semantic modeling for cloud computing, part 1. *IEEE Internet Computing*, 14(3) :81–83, 2010.

[111] A. P. Sheth and A. Ranabahu. Semantic modeling for cloud computing, part 2. *IEEE Internet Computing*, 14(4) :81–84, 2010.

[112] U. Srivastava, K. Munagala, J. Widom, and R. Motwani. Query optimization over web services. In *VLDB*, pages 355–366, 2006.

[113] M. Stal. Web services : Beyond component-based computing. *Commun. ACM*, 45(10) :71–76, Oct. 2002.

[114] D. Suciu, D. Olteanu, C. Ré, and C. Koch. *Probabilistic Databases*. Synthesis Lectures on Data Management. Morgan & Claypool Publishers, 2011.

[115] C. Szyperski. *Component Software : Beyond Object-Oriented Programming*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2nd edition, 2002.

[116] D. A. Taylor. *Object Technology (2Nd Ed.) : A Manager's Guide*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1998.

[117] S. Thakkar, J. L. Ambite, and C. A. Knoblock. A data integration approach to automatically composing and optimizing web services. In *In Proceedings of the ICAPS Workshop on Planning and Scheduling for Web and Grid Services*, 2004.

[118] T. T. L. Tran, C. A. Sutton, R. Cocci, Y. Nie, Y. Diao, and P. J. Shenoy. Probabilistic inference over rfid streams in mobile environments. In *ICDE*, pages 1096–1107, 2009.

[119] J. D. Ullman. *Principles of Database and Knowledge-base Systems, Vol. I*. Computer Science Press, Inc., New York, NY, USA, 1988.

[120] R. Vaculín, H. Chen, and K. Sycara. Modeling and discovery of data providing services. In *ICWS*, pages 54–61, 2008.

[121] W3C. W3c working group note. http ://www.w3. org/TR/ws-gloss/., 2004.

[122] D. M. Yale and D. Mcdermott. Estimated-regression planning for interactions with web services. pages 204–211. AAAI Press, 2002.

[123] W. Zhang, X. Lin, J. Pei, and Y. Zhang. Managing uncertain data : Probabilistic approaches. In *Web-Age Information Management, 2008. WAIM '08. The Ninth International Conference on*, pages 405–412, July 2008.

[124] W. Zhao, C. Liu, and J. Chen. Automatic composition of information-providing web services based on query rewriting. *SCIENCE CHINA Information Sciences*, 55(11) :2428–2444, 2012.

[125] L. Zhou, H. Chen, Y. Zhang, and C. Zhou. A semantic mapping system for bridging the gap between relational database and semantic web. In *AAAI Spring Symposium : Semantic Scientific Knowledge Integration*, pages 122–, 2008.