Les web services sémantiques, automate et intégration

Mohand-Said Hacid, Freddy Lécué, Alain Léger, Christophe Rey, Farouk Toumani

Les Web Services Sémantiques : Automates et Intégration

Partie 1 : Introduction – Motivations et Historique – Eléments et Scénario – Découverte de services web en Logique de Description – Vers la Composition de services web

Mohand-Said Hacid*, Freddy Lécué, Alain Léger**, Christophe Rey***, Farouk Toumani*****

** Universite Claude Bernard Lyon 1*
LIRIS - UFR d'Informatique Batiment Nautibus
43, boulevard du 11 novembre 1918
69622 Villeurbanne cedex, France
mshacid@iris.univ-lyon.fr

** *France Telecom R&D, 14 rue du clos courtel, F-35512 Cesson-Sévigné, France*
freddy.lecue@orange-ftgroup.com
a-t.leger@wanadoo.fr

*** *LIMOS, Université Blaise Pascal, Clermont-Ferrand, France*
{rey,ftoumani}@isima.fr

RÉSUMÉ. *Les avancées récentes des technologies « NTIC » (Nouvelles Technologies de l'Information et des Télécommunications) ont (ré)-ouvert la voie à de nouvelles solutions pour le développement d'architectures de traitement distribué dont l'impact industriel et commercial semble cette fois très prometteur. En effet, la conjonction de la maturité des architectures logicielles à composants distribués, la très forte pénétration des technologies du Web et enfin la forte pression de l'économie mondiale ont conduit à une irrésistible prolifération de composants physiques ou logiciels autonomes disponibles et distribués sur le web. Le W3C les définit comme des « services web » : « le service web est un système logiciel conçu pour rendre interopérable la communication de Machine à Machine connectées en réseaux. Il est décrit par une interface (WSDL) et peut interagir avec d'autres systèmes par l'envoi de messages (SOAP) typiquement véhiculés par un protocole HTTP et une*

représentation syntaxique XML conjointement à d'autres standards du Web ». Une très importante fonctionnalité de ces systèmes à base de services web est leur capacité à offrir des services composites construits à partir de ces composants physiques ou logiciels autonomes et réutilisables, formant une nouvelle infrastructure de traitement informatique distribué sur le Web. Profitant de l'infrastructure du web sémantique, les services web dits « sémantiques » ouvrent des perspectives d'avancées clés dans le traitement d'information couvrant un large spectre d'applications comme les services commerciaux (e-Enterprise, e-Business), la recherche scientifique (ex. grille de calcul « e-Science »), l'éducation (e-Learning) et enfin les services aux citoyens (e-Government, e-Democracy). Nous présentons dans ce tutoriel (en deux parties) un état de l'art synthétique des langages et technologies associés aux services web sémantiques, un panorama détaillé des raisonnements de découverte existants et des solutions majeures pour la composition de services web, pour terminer sur des exemples d'applications et une feuille de route.

ABSTRACT. Recent advances in networks, information and computation grids, and WWW have resulted in the proliferation of a multitude of physically distributed and autonomously developed Web services. The W3C Web Services Architecture defines ``Web service as a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine processible format (specifically WSDL) and other systems can interact with it in a manner prescribed by its description using SOAP messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards''. Correspondingly, the construction and deployment of composite services by combining and reusing independently developed component services is an important capability in the emerging Web-based computing infrastructure. Used according to the semantic web principles, semantic web services offer key IT capabilities in businesses as well as in scientific research, and key enabler of broad variety of applications including e-Enterprise, e-Business, e-Government, and e-Science. In this tutorial (in two parts) we present a concise state-of-art of languages and technologies associated to semantic web services, a detailed overview of existing discovery reasonings and key solutions for composition, and lastly typical applications and a technology roadmap.

MOTS-CLÉS : Services web, services web sémantiques, web sémantique, Architecture orientée service, découverte, composition et orchestration de services web, processus distribués, intelligence artificielle.

KEYWORDS: Web services, Semantic web, Service oriented architecture, discovery, composition and orchestration of web services, distributed process, artificial intelligence

1. Introduction

1.1. Objectifs majeurs

L'intégration d'application est l'élément clé de l'évolution actuelle des systèmes d'information et des processus en interne à l'entreprise et lors d'entreprise étendue à son réseau de partenaires (e-Business). En effet, le développement de toute nouvelle application ou d'une procédure commerciale requiert une intégration *manuelle et ad hoc* des sources de données issues d'applications et de bases de données très hétérogènes. La complexité de l'intégration est variable mais elle n'en reste pas moins une des sources de coût des plus lourdes pour les entreprises.

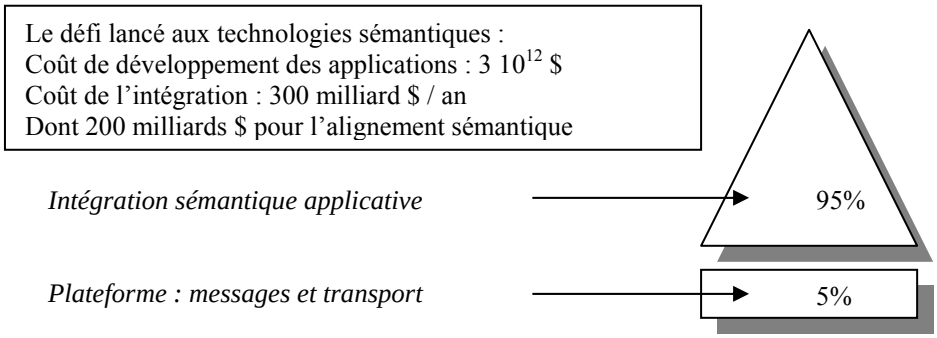


Figure 1. *Coût de l'intégration manuelle des applications*[1]

Dans ce contexte de réduction des coûts d'intégration, la technologie des Services Web propose une solution élégante à l'accès et l'intégration de ressources distribuées et hétérogènes.

L'objectif des services web est d'aller au-delà des limites de l'intégration telle qu'elle se fait actuellement dans les technologies existantes d'intégration d'applications (comme par exemple les EAI). En effet ces dernières induisent en général un grand coût de mise en œuvre, ce qui en restreint l'usage aux grandes entreprises. Or, avec l'explosion et l'ouverture apportées par Internet et l'arrivée massive de petites entreprises dans le commerce en ligne, ce n'est clairement plus adapté. Ainsi, le souhait est de passer à des technologies d'intégration décentralisées où chaque partenaire possède ses propres composants dans un système d'intégration faiblement couplé qui autorise une plus grande flexibilité dans les relations inter partenaires. Les services web permettent cette évolution (Figure 2).

[1] Sources Gartner et "CIO's guide to Semantics, version 2", Semantic Arts Inc., 2005

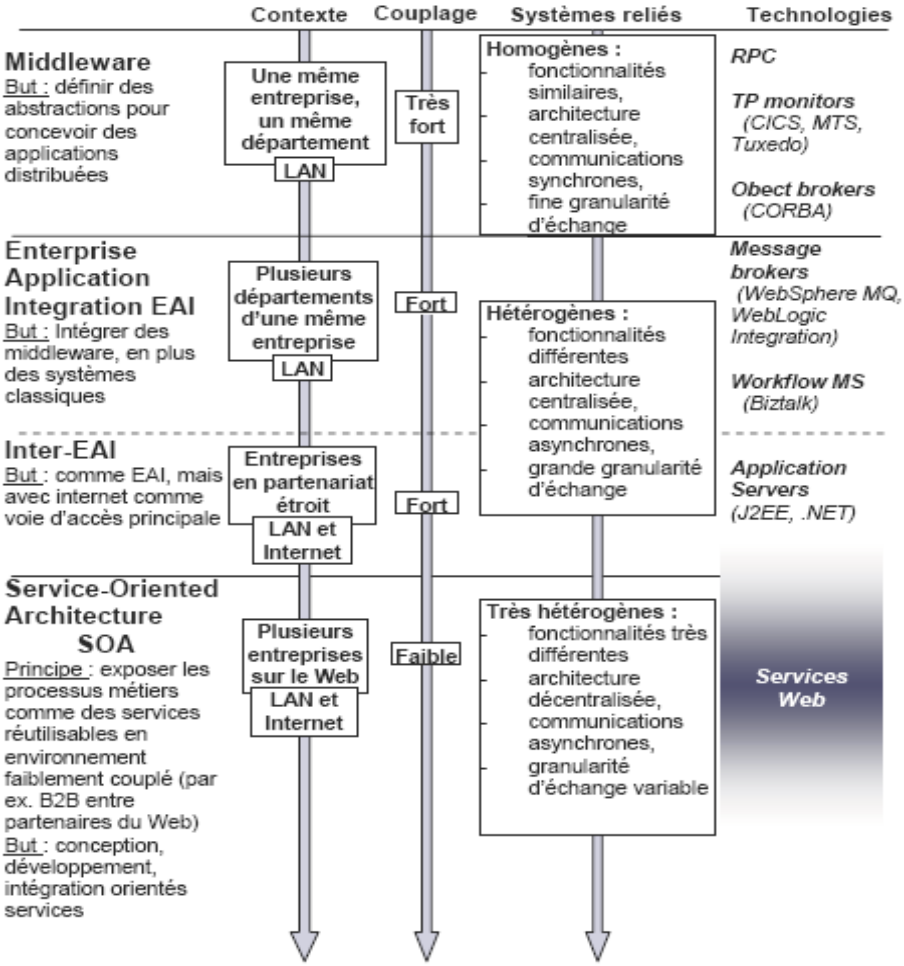


Figure 2. : *Evolution des technologies d'intégration d'applications*

Plusieurs définitions très proches ont été données des *services web* mais la référence la plus aboutie est celle du W3C[2]. En résumé, *ce sont des composants logiciels qui fournissent des services distants, distribuées et accessibles via des protocoles standardisés du Web* (W3C, OASIS). Ils peuvent fournir des informations statiques ou dynamiques et modifier les états des systèmes dans lesquels ils sont immergés.

[2] Haas and Brown 2004, W3C Web services Glosary http://www.w3.org/TR/ws-gloss/

Pour exemples très illustratifs, nous trouvons de manière classique i) le service web d'agence de voyages qui fournit une liste de vols disponibles et leurs coûts étant donnés la ville de départ, la ville d'arrivée, les dates d'aller et retour[3] ii) le service web d'achat et d'envoi de livres étant donnés le titre et un moyen de paiement[4]. Déjà sur ces exemples le paiement s'effectue généralement via un service bancaire associé ainsi que la livraison qui est assuré par un service tiers de logistique.

Les problématiques associées aux services web sont nombreuses. On peut citer la découverte (i.e. trouver des services correspondant à une requête utilisateur), la sélection (i.e. identifier le meilleur service parmi ceux découvert), la composition (i.e. construire un nouveau service à partir d'une combinaison de services existants), l'orchestration et la chorégraphie (i.e. coordonner une combinaison de services), la supervision (i.e. suivre l'exécution d'un service) , la simulation (i.e. tester avant déploiement), la vérification (i.e. s'assurer qu'un service va bien faire ce qu'il est censé faire) et le traitement d'exception (i.e. pouvoir gérer des erreurs au cours de l'exécution d'un service). La figure 3 en donne une illustration de scénarios prototypes.

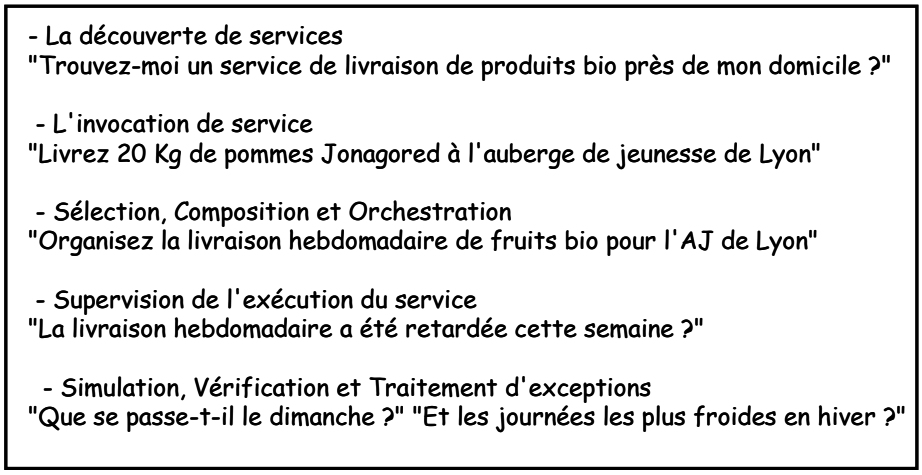


Figure 3. : *Scénarios stéréotypes*

Toutefois, cette vision de l'intégration automatisée, ad hoc et sans couture n'est possible qu'au prix d'une modélisation formelle, sémantique et comportementale des interfaces de ces « services ». C'est ainsi qu'a émergé naturellement l'extension Web Service Sémantique (SWS) des services web ("non sémantiques") actuellement

[3] Exemples "manuels" http://www.govoyages.com/ , http://www.fr.lastminute.com/
[4] Exemples "manuels" http://www.amazon.fr , http://www.priceminister.com/

déployés (HTTP[5], SOAP[6], WSDL[7]), rendant possible cette fois l'intégration et l'interopérabilité via des processus automatisés entre systèmes informatiques distribués.

C'est à cette fin que l'on voit émerger aujourd'hui des architectures et des méthodologies « orientée services » (SOA, SOC, MDA)[9] répondant d'une part à la réduction des coûts d'intégration pour les entreprises mais aussi d'autre part en anticipant sur les besoins du grand public vers une offre de services mobiles, ubiquitaires, volatiles et dynamiques.

L'utilisation des services web au sein des entreprises commence à devenir une réalité. Cependant leur usage est souvent réduit à celui de « wrappers », réalisant l'interface et la traduction de messages entre applications. La question est donc de savoir si les services web peuvent automatiser l'intégration d'applications pour rendre celle-ci dynamique, c'est-à-dire exécutable au moment voulu.

S'inspirant de la vision du web sémantique (Tim Berners-Lee et al., 2001), qui consiste à décrire les informations du Web d'une manière compréhensible par une machine afin d'automatiser des traitements fastidieux et sources d'erreurs, l'idée de services web sémantiques a émergé naturellement (Trastour et al., 2001) (McIllraith et al., 2001). Il s'agit d'enrichir les descriptions de services par l'utilisation d'une représentation sémantique formelle permettant de décrire l'offre du service et de son contexte d'usage, afin de manipuler, automatiquement, les services par raisonnement (i.e. raisonnement logique ou "inférence") à partir de cette représentation sémantique. Ajouter une représentation sémantique aux services signifie (i) les décrire à l'aide d'un langage qui possède une sémantique formelle (comme par exemple la logique du premier ordre et sa sémantique ensembliste), et (ii) utiliser des termes qui sont définis dans des ontologies standard à l'aide du même langage.

Ainsi, les technologies sémantiques appliquées au Web ont dans un premier temps visées à construire un "web de données et documents" que les applications pourront partager sur Internet pour faciliter leur interopérabilité (W3C, 2001). La même idée s'est prolongée dans un deuxième temps vers la construction d'un "web de services" que des applications pourront intégrer et partager grâce à des standards de modélisation et de représentation (OASIS, W3C, OMG, 2002). L'impact potentiel envisagé par ces approches pour des applications commerciales des entreprises, couvre les champs suivants :

- Intégration sémantique en B2B pour le partage des données et services entre entreprises partenaires, mais également pour fiabiliser et sécuriser les processus

---

[5] HTTP : http://fr.wikipedia.org/wiki/Hypertext_Transfer_Protocol
[6] SOAP : http://www.w3.org/TR/soap/
[7] WSDL : http://www.w3.org/TR/wsdl
[9] Service Oriented Architecture – Computing, Model Driven Architecture

métiers de partenariats (e.g. remplacement d'un partenaire momentanément défaillant en B2B; protocole d'échange et de négociation, etc.).

- Interopérabilité sémantique grâce aux métadonnées et aux ontologies qui supportent l'échange flexible et dynamique de données et de services entre SI d'entreprises ou unités d'affaires. Elle permet aussi de développer des systèmes d'interrogation de données, de découverte et d'intégration de services automatisée et très fiable.

- La gestion des cycles de vie des processus métiers des entreprises où les métadonnées, les ontologies et les règles deviennent des outils incontournables pour la modélisation et le traitement des données, des processus, des règles commerciales et finalement des savoirs de(s) entreprise(s).

Actuellement, les technologies sémantiques n'ont pénétré que de façon marginale le marché des NTIC. Les Technologies de l'Information traditionnelles et l'informatique procédurale dominent. Le marché et la base déployée sont immenses, et estimés à 1.2 Tera € pour le matériel, logiciels et les services.

Premier d'une série de deux, cet article présente en section 2 des éléments clés de l'approche services web sémantiques : notion de services, description d'un service, les fonctions types d'une SOA, la découverte de services web. Dans la section 3, nous donnons une vision d'ensemble détaillée de la question de la découverte de services web sémantiques dans le contexte spécifique des logiques de description. Puis après une vue d'ensemble, nous terminons par une transition vers la composition de services web, objet de la deuxième partie.

Dans la deuxième partie de cet article constituant ce panorama, nous abordons la question de la composition de services web et nous développons deux cas d'utilisation emblématiques de l'utilisation des services web. Enfin nous concluons par une feuille de route recherche et technologique sur ces technologies.

2. Descriptions et découverte de services web sémantiques

Tout comme aujourd'hui un document sur Internet (ou une bibliothèque) nécessite une indexation, une ressource de type "services web" requiert une description de son "offre de services". En effet, tout usage ou traitement relatif à un service doit pouvoir être exécuté à partir de la seule description du service, car celle-ci doit suffire à sa mise en œuvre. Dès lors la description du service occupe une place centrale dans son utilisation. C'est pourquoi il en résulte que la description d'un service est riche de plusieurs types d'informations différentes. Au niveau des services web sémantiques, cette description doit être exprimée dans un langage utilisant une sémantique claire i.e. non ambiguë et en utilisant des termes ayant eux-mêmes une définition précise dans ce même langage au sein d'une ontologie.

Dans cette section, nous présentons les différentes sémantiques des services web et les différents types de description qui en découlent. Nous positionnons les problématiques associées aux services web sémantiques par rapport à ces descriptions. Enfin, nous présentons en détails le cas de la découverte de services web sémantiques afin de pouvoir nous focaliser sur les raisonnements de découverte à la section 3.

2.1 Description des services web

Les descriptions de services se distinguent par la sémantique qu'elles induisent pour le service associé. Nous nous focaliserons sur les sémantiques ensemblistes qui permettent de définir rigoureusement chaque type de description, de pouvoir les comparer facilement, et d'être en accord avec des formalismes logiques de raisonnement basés sur ce type de sémantique. Nous distinguerons 4 types de description. Pour chacun d'entre eux nous expliquerons ce qu'il décrit du service, donnerons sa sémantique associée et dirons avec quel type de connaissances on le définit. Nous résumerons ensuite ces 4 types avec le schéma de la figure 4 et nous les illustrerons sur un exemple à la figure 5[11].

La *description abstraite* (ou conceptuelle) d'un service, regroupe les informations décrivant la fonctionnalité du service par rapport au domaine d'activité associé, c'est-à-dire la tâche qu'il réalise. Pour ce niveau de description on utilise deux représentations conjointes : un texte en langue naturelle et une représentation formelle. La description en langue naturelle ("description naturelle") utilise des mots-clés ou un texte descriptif. La sémantique associée est alors celle du langage naturel, i.e. sensible à de multiples variations interprétatives. Les mots clés ou les termes employés dans le texte font en général référence à une taxonomie ou un thésaurus décrivant le domaine du service. Une "description formelle" utilise un formalisme rigoureux comme par exemple une logique de description (LD). Sémantiquement, elle est interprétée comme l'ensemble des buts et ou des effets du service abstrait. Les concepts utilisés dans un service abstrait formel sont en général issus d'une ontologie regroupant les définitions logiques des concepts du domaine. La description abstraite est très répandue et n'est pas spécifique aux services web sémantiques. On peut citer notamment la description par mots-clés dans UDDI[12]. La description abstraite formelle est utilisée par exemple dans (Li *et al,* 2003), (Rey, 2004), (Baader *et al,* 2005), (Colucci *et al,* 2005), (Rey, 2007).

La *description fonctionnelle* d'un service est la description du service vue comme une « boîte noire » : la description fonctionnelle détaille les entrées les sorties, les pré-conditions et les effets associés au service. Elle peut être complétée par des propriétés non fonctionnelles (par exemple temps de réponse du service, éventuelles contraintes,…). On distingue les descriptions fonctionnelles

[11] Adapté et complété de figure 3 chapitre 12 de (Keller et al., 2007)
[12] http://www.oasis-open.org/committees/uddi-spec/tcspecs.shtml#uddiv3 http://uddi.xml.org/

paramétriques des descriptions fonctionnelles instanciées. Une description paramétrique décrit les entrées, sorties, pré-conditions et effets sous la forme d'un concept alors qu'une description fonctionnelle instanciée les décrit avec une valeur précise. La sémantique d'une description paramétrique est celle d'une application d'un ensemble d'états du monde vers un autre. La sémantique d'une decription fonctionnelle instanciée est celle d'une application d'un état du monde vers un autre. Dans les 2 cas, on utilise une ontologie du domaine qui peut être semblable aux ontologies utilisées pour la description abstraite. La description fonctionnelle (paramétrique ou instanciée) est utilisée par exemple dans (Paolucci *et al,* 2002), (Benatallah *et al,* 2003), (Baader *et al,*), (Lécué *et al,* 2006), (Keller *et al,* 2007).

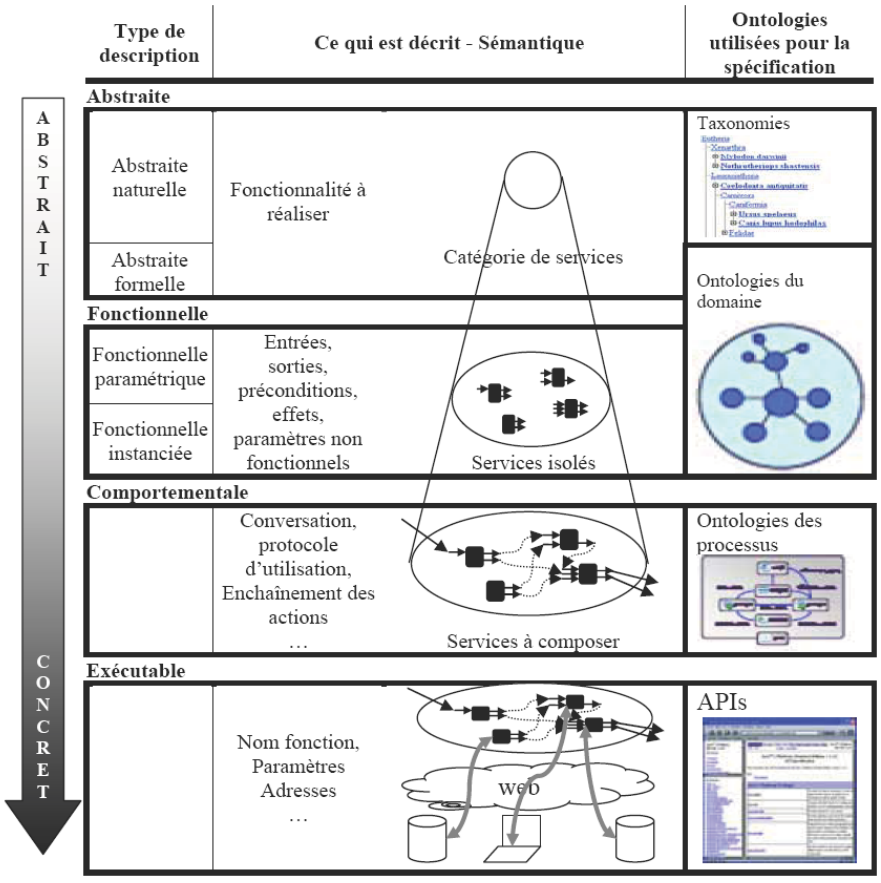


Figure 4. : *Types de* ***descriptions***

La *description comportementale* d'un service, explicite le comportement dynamique du service en décrivant le protocole de dialogue ou encore sa politique de contrôle d'états. C'est une description du service qui contient ainsi les actions à réaliser, leur enchaînement, quels sont les flux de données échangées,… Cette description doit pouvoir servir de plan d'exécution du service ainsi que de vue dynamique des points de dialogue avec le service. La sémantique de cette description est donc opérationnelle. En termes ensemblistes, c'est l'ensemble des évolutions possibles des états du service. L'ontologie utilisée est une ontologie des processus et des actions. Certains concepts utilisés dans cette ontologie peuvent être issus de l'ontologie du domaine. La description comportementale est utilisée dans (McIlraith *et al.*, 2003) (Bultan *et al.*, 2003) (Berardi *et al.*, 2003) (Pistore *et al.*, 2005a) (Berardi *et al.*, 2005b) (cf. références Partie 2).

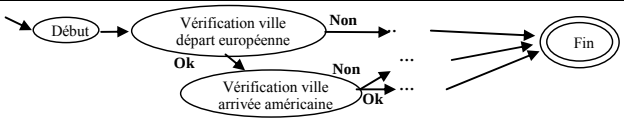
Description abstraite

naturelle	**Liste de mots clés :** RéservationBilletAvion, Vol, TransportAérien, …
formelle	**Une expression en logique de descriptions voulant dire :** le service T1 est décrit comme un vol qui arrive dans une ville, pour lequel les escales ont une durée maximale à 2h avec au moins un magasin dutyfree (à chaque escale) et qui n'accepte les animaux que s'ils sont en cage. $T_1 \equiv Vol \sqcap \forall lieuArr.Ville \sqcap \forall escale.\forall duree. \leq 2heure \sqcap$ $\forall escale.\forall mag. \geq 1DutyFree \sqcap$ $\forall escale. \geq 1mag \sqcap \forall aniAccept.\forall transport.Cage$

Description fonctionnelle

paramétrique	Le service t1 appartient à la catégorie des services décrits par le service abstrait T1. **Les entrées de t1 sont :** une ville européenne pour le départ, une ville américaine pour l'arrivée, un numéro de compte et un nom de passager. **Les sorties de t1 sont :** un numéro d'avion, un numéro de place et un prix de billet. **Les préconditions sont :** avoir un visa américain ou européen et un compte solvable. **Les effets sont :** débit de la somme sur le compte, si visa européen alors une carte de bienvenue aux Etats-Unis est donnée, et si visa américain alors un trajet en taxi est offert.
instanciée	Le service t1' est une instance de t1. **Les entrées de t1' sont :** Paris, Fort Myers, 012548E, M. Durant **Les sorties de t1' sont :** 45688zz23, 121, 450€ **Les préconditions sont :** M. Durant est américain. M. Durant a un compte solvable. **Les effets sont :** Le compte de M. Durant a été débité de 450€. Un trajet en taxi gratuit est offert à M. Durant.

Description comportementale



Description exécutable

	URL : http://www.vol_aérien.com/ … Opération : book_flight() Paramètres : String, String, int, String Valeur de retour : BookInfo

Figure 5. : *Types de* ***descriptions - exemple***

La *description exécutable* d'un service, ou « service exécutable » est la description de son interface d'appel (nom du service, nom de la procédure invocable

à distance, paramètres, adresse). La sémantique d'un service exécutable est d'être l'ensemble des appels possibles de ce service. Les connaissances utiles pour spécifier un service exécutable sont des connaissances liées à la programmation (types, API,…). Cette description est typiquement celle contenue dans le fichier WSDL associé à un service web.

2.3 Fonctions d'une architecture générique SOA

Cette section décrit les principales fonctions nécessaires à une infrastructure logicielle pour le développement et le déploiement de services Web sémantiques (SWS[13]) fortement orientée services (SOA). Ces fonctions considérées comme éléments structurels d'une infrastructure SWS par l'OASIS[14], permettent d'offrir les fonctionnalités de base suivantes : description, découverte, adaptation, sélection, médiation, composition, chorégraphie, orchestration et exécution de services Web sémantiques (Knowledge Web Del 1.2.5 v2, Architecture of the Semantic Web Framework v2, 2007).

A ce point de l'exposé, il est bon de désembrouiller une confusion fréquente sur les notions de "service", "services web" et "applications", et à ce propos de signaler le très remarquable effort réalisé par l'OASIS pour définir précisément la terminologie et les éléments clés d'une architecture SOA[15]. Ce document décrit une architecture générique indépendante de tout choix de langages, protocoles, technologies et plateformes. Le modèle définit un SOA comme *un paradigme pour organiser et utiliser des capacités distribuées qui peuvent-être sous le contrôle de plusieurs domaines de propriété*. Ainsi la SOA fournit une puissante architecture pour l'appariement de besoins et d'offres (ou capacités) très souvent combinées pour répondre à ces besoins. Les concepts centraux introduits par le modèle sont : service, effets réels, capacités, description de service, visibilité, interaction, contrat et directives, et enfin contexte d'exécution, supervision et conduite en cas erreurs.

La capacité est la faculté de réaliser une (ou des) action(s) ayant une valeur tangible à la réponse d'un besoin. Par exemple, une capacité de "réservation de vols" est une offre générique qui ne précise pas le moyen nécessaire pour la mettre en œuvre. Ici le client peut soit se déplacer physiquement vers l'agence de voyage de son choix ou faire appel à une application sur le web (site web de réservation) depuis son téléphone mobile ou son ordinateur personnel. *Un service est donc un mécanisme par lequel cette capacité peut-être mise en œuvre*.

La fonction de *description* de services Web permet de prendre en charge le modèle de représentation des services e.g. OWL-S (Ankolenkar *et al.*, 2004), WSMO (Lausen *et al.*, 2005), SWSF (Battle et al., 2005) ou encore SA-WSDL

---

[13] SWS : Semantic Web Services
[14] OASIS : http://www.oasis-open.org
[15] OASIS Reference Model for Service Oriented Architecture V 1.0 - Official Committee Specification approved Aug 2, 2006

(Akkiraju *et al.*, 2005). Ces derniers permettent ainsi de décrire les services Web à plusieurs niveaux d'abstraction i.e., niveau fonctionnel (SA-WSDL, OWL-S profile, WSMO capability), niveau comportemental (OWL-S process, WSML choreography, WSBPEL[16]).

La fonction de *découverte* de services Web est responsable d'une recherche spécialisée sur la base de la description sémantique de leur interface. Ainsi le moteur de découverte (en fait simplement un moteur de recherche en ressources de type « services web ») va extraire de l'ensemble de l'offre de services Web[17] les services qui répondent « le mieux » aux besoins de cette recherche. « le mieux » tiendra compte de contraintes de typage sémantique (e.g. compatibilité des données échangées) mais aussi de contraintes de contextes d'usage plus globaux (e.g, Qualité de Services (QoS), contexte d'utilisation du service, stratégie de l'utilisateur, politique privée d'usage).

A l'instar du tri opéré par l'utilisateur humain sur les réponses d'un moteur de recherche sur le web, la *sélection* de services Web trie les plus pertinents. Ce composant aussi dénommé « composant de négociation » est responsable de la vérification et du filtrage des services découverts au moyen de conditions et de contraintes d'utilisation. De plus, un tel modèle de sélection permet d'établir un classement de ces services en fonction de leur adéquation aux besoins.

La fonction de *composition* de services permet de créer de nouveaux services à partir de services existants dans le but d'obtenir une nouvelle fonctionnalité. Cette nouvelle fonctionnalité est obtenue par composition à deux niveaux i.e. fonctionnel et comportemental, en prenant en compte les contraintes liées au typage sémantique de la nature des messages à échanger et à la chorégraphie (i.e. interactions dialogiques, échange de messages) des services à assembler.

La fonction alignant les *chorégraphies* entre services Web permet de faciliter la conversation entre utilisateurs et fournisseurs de services. La partie chorégraphie est donc capable d'implémenter un langage facilitant les échanges de messages et interactions entre services, mais aussi de prendre en compte les contraintes liées aux comportements des services Web. Ce composant a pour but de faciliter l'exécution des interactions des services web composant cette chorégraphie.

L'*orchestration* des services Web est fortement dépendante des traitements de composition, chorégraphie et médiation. Cette fonction assure l'enchaînement global des services composés en tenant compte des contraintes d'ordonnancement de ceux-ci mais aussi des réconciliations définies par le médiateur. Cette orchestration peut être représentée au moyen de différents standards (WSBPEL) ou proposition de standards e.g. OWL-S (Ankolenkar *et al.*).

---

[16] OASIS : http://www.oasis-open.org puis WS-BPEL

[17] Qui pourrait être très large et ouverte sur le web. A ce titre un annuaire global (UDDI) a été proposé pour enregistrer les services web comme on le ferait pour un annuaire de personnes

Et globalement, pièce maîtresse de cette architecture, quelque part implanté sur l'ESB on trouvera nécessairement une fonction de *médiation* associée à une (ou des) ontologie(s)[18] décrivant les termes (et leurs relations) avec lesquels les services communiquent et qui permettra de réconcilier l'hétérogénéité inévitable lors "du faire ensemble" de ces composants et services, tant au niveau conceptuel, fonctionnel que comportemental. Plus précisément, par exemple la médiation au niveau fonctionnel permettra la transformation d'un format ou d'un typage de données, la médiation de niveau comportemental tentera de réconcilier les différences des protocoles d'interactions, de communications et d'échanges de messages[19].

2.4 Découverte de services web

Classiquement, la découverte automatique de services web se fait en 5 étapes (figure 6) : (1) les fournisseurs enregistrent leurs services auprès d'un annuaire qui est donc essentiellement un catalogue de descriptions de services, (2) un utilisateur (humain ou logiciel) pose une requête à l'annuaire afin d'identifier les services qui répondent à ses besoins, (3) un moteur de découverte de services va identifier ceux qui sont intéressants par rapport à la requête de l'utilisateur et va les renvoyer à l'annuaire, (4) l'annuaire les transmet à son tour à l'utilisateur et (5) l'utilisateur est en mesure d'exécuter les services à distance directement auprès des fournisseurs.

L'automatisation de la découverte de services web est une des étapes importantes pour permettre une intégration qui tire profit de la décentralisation et de la flexibilité du web. Cependant, il ne faut pas oublier que ce scénario classique de découverte est basé sur les descriptions textuelle et exécutable des services web, i.e. soit sur des mots clés et/ou une description en langage naturel, soit sur les interfaces des procédures à appeler. Dans le cas de l'intégration B2B, on peut se demander si cela est suffisant. Autrement dit, est-il envisageable qu'une entreprise délègue à un processus de découverte automatique le soin de rechercher un partenaire pour une mission précise sur l'unique base de mots-clés et de prototypes de procédures ?

Dans ce cas, les mécanismes classiques de découverte sont des recherches par mots clés liées à des techniques de recherche d'informations basées par exemple sur des notions de similarité linguistique ou de "clustering" de ces mêmes mots clés. Bien entendu, la présence d'un annuaire de service permet aussi un autre cas très courant qui est la découverte par navigation manuelle dans l'annuaire avec visualisation des services ordonnés selon divers critères. Dans les deux cas (automatique et manuelle), la découverte est basée sur les descriptions abstraite

---

[18] Une modélisation formelle et consensuelle d'un domaine [Gruber 95]

[19] Problématique en soi, pour des raisons de place nous renvoyons le lecteur intéressé sur (WSMO http://www.w3.org/Submission/WSMO/), (Vitvar et al., 4th IEEE Conference on Enterprise Computing, E-Commerce and E-Services (EEE07) , IEEE CS Press, July, 2007), (Euzenat and Shvaiko, Ontology matching, Springer Verlag, 2007)

naturelle et exécutable des services. Si on considère un contexte limité à des partenaires déjà connus qui partagent une même terminologie de mots clés, alors cela paraît possible. Cependant l'intérêt des services web n'est alors plus évident puisque la problématique est très semblable à celle de l'intégration inter-entreprise (pour laquelle d'autres technologies performantes existent).

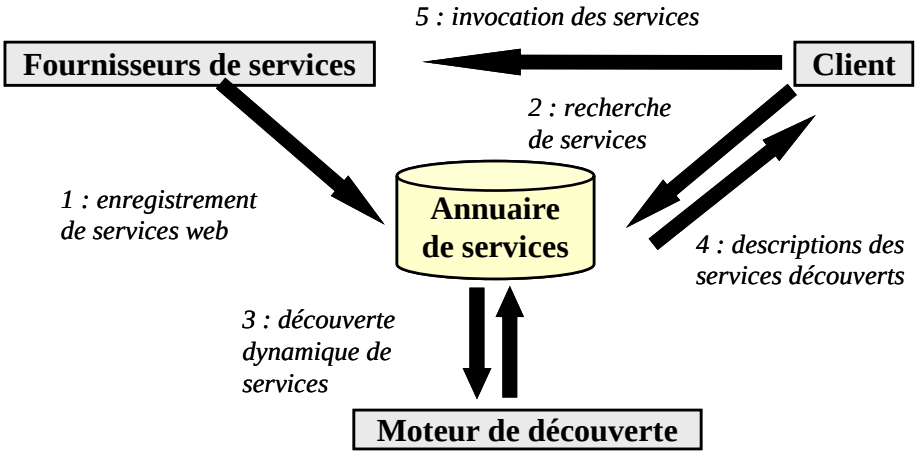


Figure 6. : *Découverte de services web - Architecture générique*

Si on considère le contexte du web, dans lequel l'entreprise n'a pas d'a priori et recherche sur le web le meilleur partenaire au moment où elle en a besoin, alors le scénario de découverte entièrement automatique n'est pas réaliste sur la seule base de mots clés et de prototypes de procédures. Sans même parler de problématiques annexes (sécurité, confidentialité, gestion transactionnelle, …), la découverte de services ne peut aboutir à des résultats suffisamment pertinents pour permettre une invocation automatique dès identification du service, sans intervention humaine. Le problème est ici essentiellement le même que celui des moteurs de recherche : à partir des mêmes techniques, soit on obtient beaucoup de réponses non pertinentes, soit on ne voit pas des réponses qui le seraient.

Ajouter une sémantique à la description des services apparaît alors comme une solution[20]. C'est parce que la sémantique ne va pas être la même suivant l'aspect du service que l'on veut décrire que l'on aboutit à plusieurs types de descriptions (Fig. 4), avec à chaque fois une sémantique précise et un formalisme adapté (logique, automates, planning,…).

A l'image de ce qui est proposé dans (Keller *et al.*, 2007), on pourrait résumer le processus de découverte en 3 étapes (figure 7) : la compréhension de la requête

[20] C'est exactement le principe du web sémantique au niveau des données du web.

utilisateur à partir de l'expression de ses besoins, la découverte de services web sémantiques à proprement parler et l'obtention de services exécutables à partir des services web sémantiques découverts. L'avantage d'un tel processus est qu'il présente les problématiques communes à la découverte sémantique quel que soit le type de description utilisée (abstraite, fonctionnelle ou comportementale).

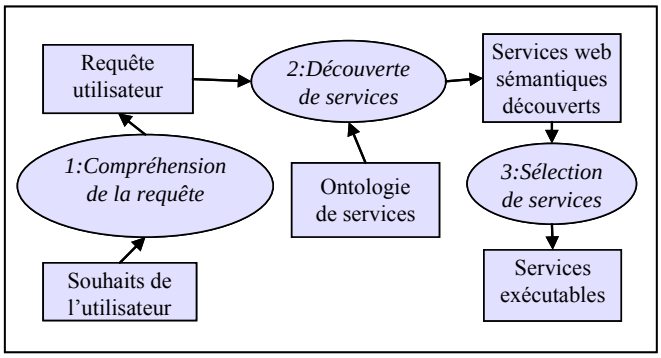


Figure 7. : *Etapes de découverte de services web sémantiques*

Exemple: l'exemple suivant montre comment s'enchaînent les 3 étapes de la découverte. L'étape 1 est la suivante : à partir d'une requête exprimée en langue naturelle, on suppose qu'on est capable d'interpréter correctement le but visé et de traduire dans une description de concept. A l'étape 2, on exécute un raisonnement de découverte qui cherche dans une ontologie de services abstraits ceux qui correspondent à la requête. Enfin, à l'étape 3, on suppose que pour chaque service abstrait de l'ontologie, on connaît plusieurs services exécutables. Ainsi pour chaque service découvert, on pourra proposer un ou plusieurs services exécutables (soit en les classant par rapport à un critère donné, soit en laissant l'utilisateur choisir).

- Etape 1 : Compréhension de la requête

  - Souhaits de l'utilisateur : « Je veux un vol pour Fort Myers avec des escales d'au plus 1h et aucun animal accepté à bord, ainsi qu'un hébergement d'au plus 2 étoiles avec piscine et écran de TV plat. »

  - Requête utilisateur abstraite: c'est un concept exprimé dans la logique de description *ALN* décrivant un vol dont le lieu d'arrivée est une ville de moins de 100 000 habitants, avec des escales limitées à 1h, où les animaux de compagnie ne sont pas acceptés, ainsi qu'un hébergement d'au plus 2 étoiles équipé d'une piscine et d'un écran plat de télévision.

$$Q \equiv \quad Vol \sqcap \forall lieuArr.Ville \sqcap \forall lieuArr. \leq 100000 hab \sqcap \forall escale.\forall duree. \leq 1heure \sqcap \forall aniAccept.\bot \sqcap Heberg \sqcap \leq 2etoile \sqcap \forall equip.Piscine \sqcap \forall tv.EcranPlat$$

- Etape 2 : Découverte de services abstraits

- Ontologie de services abstraits : on dispose d'une ontologie définissant 8 services avec la même logique de description que la requête. T_1, T_2 et T_3 sont des services abstraits de transport aérien qui décrivent 3 types de services fournissant des billets d'avion. H_1, H_2, H_3 et H_4 sont des services abstraits d'hébergement qui décrivent 4 types de services permettant de réserver un hébergement. V_1 est un service abstrait de location de véhicule.

$$
\begin{array}{ll}
Trajet : T_1 \equiv & Vol \sqcap \forall lieuArr.Ville \sqcap \forall escale.\forall duree. \leq 2heure \sqcap \\
& \forall escale.\forall mag. \geq 1DutyFree \sqcap \forall escale. \geq 1mag \sqcap \forall aniAccept.\forall transport.Cage \\
Trajet : T_2 \equiv & Vol \sqcap \forall lieuArr.Ville \sqcap \forall escale.\forall mag.BoutiqueLuxe \sqcap \\
& \forall escale.\forall mag. \leq 0DutyFree \sqcap \forall aniAccept.Chien \sqcap \forall tv.EcranPlat \\
Trajet : T_3 \equiv & Vol \sqcap \forall lieuArr.Ville \sqcap \forall lieuArr.\forall situe.France \sqcap \forall escale.\forall duree. \leq 1heure \sqcap \\
& \geq 1escale \sqcap \forall aniAccept. \leq 10kg \\
Hotel : H_1 \equiv & Heberg \sqcap \leq 2etoile \sqcap \geq 2etoile \sqcap \forall equip.Piscine \sqcap \forall tv.\forall chaine.Satellite \sqcap \\
& \forall tv. \geq 10chaine \sqcap \forall equip.Tennis \sqcap \forall aniAccept.\forall transport.Cage \\
Hotel : H_2 \equiv & Heberg \sqcap \geq 1etoile \sqcap \forall tv.\forall chaine.\neg Satellite \sqcap \forall tv.EcranPlat \sqcap \\
& \forall aniAccept. \leq 10kg \sqcap \forall aniAccept.Chien \\
Hotel : H_3 \equiv & Heberg \sqcap \forall tv.\bot \sqcap \leq 1etoile \\
Hotel : H_4 \equiv & Heberg \sqcap \leq 3etoile \sqcap \geq 3etoile \sqcap \forall equip.Piscine \\
Voiture : V_1 \equiv & Vehicule \sqcap \forall categorie.Berline \sqcap \forall motorisation.4x4
\end{array}
$$

- (Ensembles de) services web sémantiques (abstraits) découverts : le raisonnement de découverte utilisé ici est celui des meilleures couvertures d'un concept en utilisant une terminologie (voir § 3) qui renvoie tous les ensembles de services qui ont le plus d'informations communes avec la requête :

Solutions n° 1, 2 et 3 = $\{T_3, H_1, H_2\}$, $\{T_2, T_3, H_1\}$, $\{T_1, T_2, H_1, H_2\}$

- Etape 3 : Découverte de services exécutables

Correspondance entre services abstraits et services exécutables pour la solution n° 1

- à T3 correspondent T3exec= {BilletAirFrance, BilletDeltaAL}, à H1 correspondent H1exec= {HotelParis, HotelFloride} et à H2 correspondent H2exec={HotelTexas, HotelFloride}

Les services exécutables proposés à l'utilisateur pour la solution n° 1 peuvent alors être :

Soit, un triplet du produit cartésien T3exec x H1exec x H2exec choisi selon un certain critère. Par exemple, on peut choisir le premier triplet que l'on trouve ayant le plus de services en double : ici ce sera {BilletDeltaAL, HotelFloride, HotelFloride} qui peut se réduire à {BilletDeltaAL, HotelFloride}.

Soit l'ensemble de tous les triplets de T3exec x H1exec x H2exec : (8 triplets) : {BilletAirFrance, HotelParis, HotelTexas}, {BilletAirFrance, HotelParis, HotelFloride}, …. { BilletDeltaAL, HotelFloride, HotelFloride },

On laisse alors le choix à l'utilisateur.

Même déroulement pour les solutions n° 2 et 3.

On remarque ici que les triplets renvoyés en résultats ne disent pas du tout à l'utilisateur comment appeler les 3 services concrètement. On voit ainsi la nécessité de coupler la phase de découverte de services abstraits avec une phase de découverte de services fonctionnels suivie d'une phase de composition de services à partir des descriptions comportementales. On pourrait ainsi déterminer comment sont utilisées les valeurs de la requête initiale par rapport aux entrées des services (par exemple « Fort Myers » doit devenir une entrée d'un vol), et aussi dire comment enchaîner l'exécution des services entre eux quand plusieurs ont été découverts – mais c'est l'objet de la composition cf. Partie 2.

Dans la section suivante, à l'instar de l'exemple précédent, nous allons nous focaliser sur l'étape 2 de ce processus, avec des descriptions de services abstraites et formalisées avec les logiques de description. En effet, les logiques de description occupent une place centrale parmi les formalismes de représentation des connaissances et de raisonnement utilisés pour le web sémantique et pour les services web sémantiques.

Toutefois, nous avertissons le lecteur que des méthodes hybrides de mise en correspondance existent, alliant des approches de traitement de langue (TAL) sur les déclarations textuelles des fonctions et interfaces des SWS et des approches logiques, mais qui pour des raisons de place ne seront pas exposées dans le cadre de cet article, nous revoyons le lecteur intéressé aux deux excellents articles (Klusch *et al.,* 2007), (Kaufer *et al.*, 2006).

3. Etat de l'art scientifique de la découverte de services web sémantiques

3.1. Notions sur les logiques de description

Le formalisme logique sous-jacent aux raisonnements présentés est la famille des logiques de description. Bien connues comme l'un des piliers du web sémantique, les logiques de description sont bien adaptées à la description des services abstraits. En effet leur sémantique ensembliste capte naturellement l'idée de catégorie de services induite par l'expression de services abstraits. Les logiques de description sont un formalisme de représentation des connaissances et de raisonnements étudiées depuis environ 25 ans. Elles constituent une famille de sous-langages de la logique du premier ordre dont l'objectif est la modélisation de structures hiérarchiques complexes et le raisonnement sur ces structures.

Une logique de description permet de décrire l'aspect terminologique (intentionnel) d'un domaine en termes de concepts, qui sont des ensembles d'individus (i.e. des relations unaires), et en termes de rôles (i.e. des relations binaires entre individus). Chaque concept est décrit par une description de concept,

elle-même construite par l'intermédiaire de constructeurs de concepts. Parmi l'ensemble des constructeurs existants, la donnée d'un sous-ensemble définit une logique de description particulière.

Schéma	Notation	Explication
⊥	Bottom	Concept vide
T	Top	Le domaine entier, le concept le plus général (i.e. l'absence de contrainte)
A	A	Une description de concept
B	B	Un ensemble de descriptions de concepts
A / B	B ⊑ A	B est subsumé par A
A / B	B ≡A	B est equivalent à A (i.e. B est subsumé par A et A est subsumé par B)
A / B / C	A ⊓B ≡C	La conjonction de A et B est equivalente à C
A B	A ⊓B ≡ ⊥	La conjonction de A et B est vide (i.e. A et B sont disjoints)
max / A	A max p/r à ⊑	A est maximal par rapport à la subsomption (telle que certaines autres propriétés sont vérifiées)
A / B / lcs(A,B)	lcs(A,B)	Plus petit subsumant commun de A et de B (least common subsumer)

Tableau 1. : *Relations entre descriptions*

Par exemple, on pourrait formuler le service abstrait « voyage en train vers Paris » par la description suivante :

$S_1 \equiv$ Voyage ⊓ (∀ vehicule.Train) ⊓ (∀ destination.Paris) ⊓ (∀ durée.(≤ 2h))

Cette description utilise les constructeurs ∀ (quantificateur universel) et ⊓ (conjonction) définissant, avec le constructeur Top (désignant l'univers entier), le constructeur Bottom (désignant l'ensemble vide, de symbole ⊥), la négation de concept ¬, et les restrictions numériques ≥nR et ≤nR, la logique de description ALN. Ici, on définit (par le symbole de l'équivalence ≡) le service abstrait S1 comme étant la conjonction du concept Voyage et des restrictions de concepts (∀ vehicule.Train), (∀ destination.Paris) et (∀ durée.(≤2h)). Cela signifie que tout service fonctionnel instance de S1 est un service de voyage dont le véhicule est le train, la destination Paris et la durée inférieure à 2h.

On peut ainsi exprimer par des descriptions de concept les contraintes définissant les services. Par exemple, la description « ∀ destination.Paris » traduit la contrainte sur le lieu de destination. On peut ensuite regrouper ces descriptions par le constructeur conjonction ⊓ pour en former des descriptions plus complètes. Cela

traduit l'accumulation de contraintes. En termes ensemblistes, la conjonction correspond à l'intersection des interprétations des descriptions. Si on définit un autre service S_2, la conjonction $S_1 \sqcap S_2$ doit être comprise comme un service abstrait qui est un package des services abstraits S_1 et S_2. En effet, certains raisonnements de découverte cherchent à faire correspondre une demande avec des packages de services et non avec des services séparés.

La sémantique ensembliste des logiques de description induit d'éventuelles relations d'inclusion entre descriptions de concept nommées ou relations de subsomption. Ces relations de subsomption sont importantes car ce sont elles qui structurent hiérarchiquement les concepts les uns par rapport aux autres. En termes de services et de contraintes, plus une description de concept est grande par rapport à la subsomption, plus le service abstrait correspondant est vague, moins il est contraint, et donc, plus il regroupe potentiellement de services fonctionnels correspondants. Par exemple, on peut imaginer un service S_3 défini par $S_3 \equiv$ Voyage. On dira alors que S_1 est subsumé par S_3, i.e. que S_1 est plus contraint que S_3. Comme on va le voir, la relation de subsomption est à la base de presque tous les raisonnements de découverte de services web sémantiques.

Le Tableau 1 récapitule les principaux éléments des logiques de description nécessaires pour bien comprendre les raisonnements de découverte présentés par la suite. Il va de soit que cette présentation des logiques de description est très succincte. Pour une présentation plus complète des logiques de descriptions, voir (Baader *et al.*, 2003).

3.2. Découverte dans le contexte des logiques de description

Les raisonnements de découverte ont pour but la mise en correspondance (« matchmaking » ou aussi « matching ») des requêtes utilisateurs avec des services, sur la base de leur description en logiques de descriptions. Dans (Cardoso, 2007), la notion de degré de correspondance (Degree Of Match) est définie comme « une valeur qui exprime la similarité entre deux entités par rapport à une métrique de similarité donnée ». Ce degré de correspondance permet de classer les services découverts. C'est aussi ce qui permet de caractériser les raisonnements de découverte. Nous proposons d'affiner cette notion de degré de correspondance au travers des 4 caractéristiques suivantes :

– La relation : c'est la relation qui doit-être vérifiée entre les solutions et les requêtes. Par exemple, on peut vouloir des services « équivalents à » ou bien encore « subsumant » la requête.

– L'arité : c'est le nombre de services qui peuvent former les réponses mises en correspondance avec une requête. On notera "1-N" si une réponse du système de correspondance à une requête peut être construite à partir de plusieurs services (souvent une conjonction de services). Sinon, si les réponses à une requête ne sont constituées que d'un seul service, alors la découverte est notée d'arité "1-1".

– La distance : c'est la mesure de la différence qui sépare la requête des services découverts. Cette mesure peut être quantifiée numériquement ou bien exprimée sous la forme d'une description en logiques de descriptions. Cette dernière possibilité est plus riche puisqu'on peut alors travailler avec d'autres raisonnements sur cette description, pour par exemple entamer un dialogue avec l'utilisateur, transmettre la partie incomprise de la requête à un autre système de découverte de services,…

– L'ordre : c'est la manière de classer les solutions de même "arité" et vérifiant la même relation. Cet ordre est en général défini comme un ordre sur les distances respectives des solutions par rapport à la requête.

	Sans calcul de distance	**Avec calcul de distance**
Arité 1-1	• L'équivalence • La subsomption • Plug-in • L'intersection (non vide) • Le super concept • La disjonction (intersection vide)	• L'abduction • La contraction
Arité 1-N	• L'équivalence • La subsomption maximale (« maximal containment »)	• Meilleure couverture de concept en utilisant une terminologie • Couverture par abduction

Tableau 2. : *Les différents raisonnements de découverte classés par catégories*

Ainsi, à partir d'une requête Q et d'un ensemble T= $\{S_i\}$ de services, un raisonnement de découverte est un raisonnement qui :

– recherche toutes les descriptions E qui :

- vérifie une certaine relation avec Q et

- vérifie une certaine arité (si l'arité est 1-1, alors chaque E est un service S_i de T, si l'arité est 1-N, alors chaque E est une conjonction de S_i de T)

– classe éventuellement ces descriptions E par rapport à un certain ordre

– donne éventuellement une mesure de distance de E par rapport à Q.

Le Tableau 2 présente une classification des raisonnements de découverte en 4 catégories : d'arité 1-1 ou 1-N, avec ou sans calcul de distance. La section suivante reprend en détails chacun des raisonnements existant dans ces 4 catégories.

3.3 Les différentes catégories de raisonnements de découverte dans les LD

Raisonnements 1-1 sans calcul de distance

Le Tableau 3 s'intéresse aux raisonnements d'arité 1-1 qui ne calculent aucune distance entre Q et E. Dans le raisonnement d'équivalence, chaque S_i découvert

(constituant à chaque fois une solution E) doit être exactement équivalent à Q. Autrement dit, l'utilisateur a formulé (peut-être avec d'autres termes) une demande qui est exactement la même proposition que le service S_i. Dans le raisonnement de subsomption, S_i est découvert quand S_i répond à la même demande que Q mais avec des contraintes additionnelles. Dans le raisonnement plug-in, S_i est découvert quand S_i répond à une version moins contrainte de Q. Dans le raisonnement par intersection, S_i sera découvert s'il répond partiellement à Q. Dans le raisonnement par super concept, S_i est découvert quand il existe un service non trivial (i.e. différent de Top) subsumant S_i et Q, même s'il n'y a pas d'intersection entre les deux (i.e. si Q et S_i sont disjoints). Enfin quand il n'existe aucun S_i vérifiant une de ces relations, alors c'est un cas de disjonction.

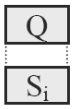
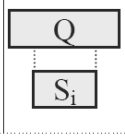
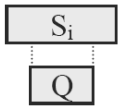
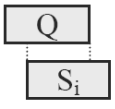
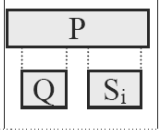
Nom	Equivalence	Subsomption	Plug-in	Intersection	Super concept	Disjonction
Relation	$Q \equiv S_i$ et $Q \sqcap S_i \not\equiv \bot$	$S_i \sqsubseteq Q$ et $Q \sqcap S_i \not\equiv \bot$	$Q \sqsubseteq S_i$ et $Q \sqcap S_i \not\equiv \bot$	$Q \sqcap S_i \not\equiv \bot$	$\exists P \neq Top$ tel que : $S_i \sqsubseteq P$ et $Q \sqsubseteq P$	$Q \sqcap S_i \equiv \bot$
Schéma						
Cf	(Gonzalez-Castillo *et al.*, 2001), (Li. L *et al.*, 2003) et (Paolucci et al., 2002)					

Tableau 3. : *Les raisonnements d'arité 1-1 sans calcul de distance*

Le Tableau 3 s'intéresse aux raisonnements d'arité 1-1 qui ne calculent aucune distance entre Q et E. Dans le raisonnement d'équivalence, chaque S_i découvert (constituant à chaque fois une solution E) doit être exactement équivalent à Q. Autrement dit, l'utilisateur a formulé (peut-être avec d'autres termes) une demande qui est exactement la même proposition que le service S_i. Dans le raisonnement de subsomption, S_i est découvert quand S_i répond à la même demande que Q mais avec des contraintes additionnelles. Dans le raisonnement plug-in, S_i est découvert quand S_i répond à une version moins contrainte de Q. Dans le raisonnement par intersection, S_i sera découvert s'il répond partiellement à Q. Dans le raisonnement par super concept, S_i est découvert quand il existe un service non trivial (i.e. différent de Top) subsumant S_i et Q, même s'il n'y a pas d'intersection entre les deux (i.e. si Q et S_i sont disjoints). Enfin quand il n'existe aucun S_i vérifiant une de ces relations, alors c'est un cas de disjonction.

Exemple : Nous reprenons la partie de la requête de l'exemple courant restreinte à la demande d'un hébergement d'au plus 2 étoiles avec piscine et télévision sur

écran plat. Et on suppose une ontologie du domaine du tourisme définissant 3 concepts, une requête Q_2 et une offre de 11 services S_i :

$$HebergHauteGamme \equiv Heberg \sqcap \geq 3etoile$$
$$HebergMoyenneGamme \equiv Heberg \sqcap \leq 2etoile$$
$$HotelMoyenneGamme \equiv Hotel \sqcap HebergMoyenneGamme$$

$$Q_2 \equiv Heberg \sqcap \leq 2etoile \sqcap \forall equip.Piscine \sqcap \forall tv.EcranPlat$$

$$S_1 \equiv HebergMoyenneGamme \sqcap \forall equip.Piscine \sqcap \forall tv.EcranPlat$$
$$S_2 \equiv HotelMoyenneGamme \sqcap \forall equip.Piscine \sqcap \forall tv.EcranPlat$$
$$S'_2 \equiv HebergMoyenneGamme \sqcap \forall equip.(Piscine \sqcap Tennis) \sqcap \forall tv.EcranPlat$$
$$S_3 \equiv Heberg \sqcap \forall equip.Piscine \sqcap \forall tv.EcranPlat$$
$$S'_3 \equiv HebergMoyenneGamme \sqcap \forall equip.Piscine$$
$$S_4 \equiv Heberg \sqcap \forall equip.(Piscine \sqcap Tennis) \sqcap \forall tv.EcranPlat$$
$$S'_4 \equiv HebergMoyenneGamme \sqcap \forall equip.Piscine \sqcap \forall tv.CRT$$
$$S_5 \equiv HebergHauteGamme \sqcap \forall equip.Piscine \sqcap \forall tv.EcranPlat$$
$$S'_5 \equiv HebergMoyenneGamme \sqcap \geq 1equip \sqcap \forall equip.(\neg Piscine \sqcap Tennis) \sqcap \forall tv.EcranPlat$$
$$S_6 \equiv CentreLoisirs \sqcap \neg Heberg \sqcap \leq 2etoile \sqcap \forall equip.Piscine \sqcap \forall tv.EcranPlat$$

On aura alors les relations suivantes entre les services et la requête :

Equivalence $Q \equiv S_i$	S_1	D'après la définition de HebergMoyenneGamme.
Subsomption $S_i \sqsubseteq Q$	S_2 S'_2	Pour S_2 : car HotelMoyenneGamme est subsumé par HébergementMoyenneGamme. Pour S'_2 : car on rajoute une contrainte concernant le terrain de Tennis.
Plug-in $Q \sqsubseteq S_i$	S_3 S'_3	Pour S_3 : car Heberg subsume HébergementMoyenneGamme. Pour S'_3 : car la contrainte relative à la télévision n'est plus dans S'_3.
Intersection $Q \sqcap S_i \not\equiv \bot$	S_4 S'_4	Pour S_4 : car Heberg subsume HébergementMoyenneGamme mais en même temps on rajoute une contrainte concernant le terrain de Tennis. Pour S'_4 : car $\forall$ tv.EcranPlat et $\forall$ tv.CRT sont incomparables par rapport à la subsomption, sans être incompatibles (relativement à l'ontologie du domaine).
Super concept $\exists P \neq Top$ tq $S_i \sqsubseteq P$ et $Q \sqsubseteq P$	S_5 S'_5	Pour S_5 : car HebergMoyenneGamme et HebergHauteGamme sont incompatibles (relativement aux 3 définitions de l'ontologie du domaine), mais cependant Heberg est un concept de l'ontologie qui subsume les 2. Pour S'_5 : car « $\forall$ equip.Piscine » est incompatible avec « $\geq$1equip$\sqcap$ $\forall$ equip.(¬Piscine$\sqcap$ Tennis) », mais cependant Heberg est un concept de l'ontologie qui subsume les 2..
Disjonction $Q \sqcap S_i \equiv \bot$	S_6	Pour S_2 : car ¬Heberg et Heberg sont incompatibles et, à cause de CentreLoisirs, aucun concept ne subsume à la fois Q et S_6.

Raisonnements 1-1 avec calcul de distance

Le Tableau 4 s'intéresse aux raisonnements d'arité 1-1 qui calculent une distance entre Q et E. Cette distance est une description de concept qui fait le lien

entre Q et les S_i découverts, et qui permet aussi d'ordonner les différentes solutions les unes par rapport aux autres.

Dans le raisonnement d'abduction, il s'agit de découvrir les S_i ayant une intersection non vide avec Q et pour lesquels il existe une description H dont l'intersection avec S_i (cas 1), resp. avec Q (cas 2), est subsumée par Q (cas 1), resp. par S_i (cas 2). En termes de services et de requêtes, cette découverte est utilisée quand on est dans un cas d'intersection et qu'on veut se ramener à un cas de subsomption ou de « plug-in ». Ce passage s'effectue au prix de l'ajout des contraintes de H à S_i (cas 1) ou à Q (cas 2). Ce raisonnement d'abduction permet donc une découverte « négociée » : soit l'utilisateur ajoute des contraintes pour coller avec une offre, soit c'est le fournisseur qui précise son offre pour coller avec la requête.

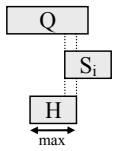
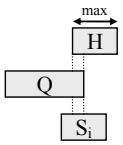
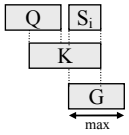
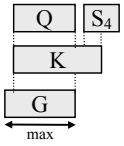
Nom	Abduction de l'intersection non vide à la subsomption (cas 1)	Abduction de l'intersection non vide au plug-in (cas 2)	Contraction de la disjonction à l'intersection non vide (cas 1)	Contraction de la disjonction à l'intersection non vide (cas 2)
Relation	$Q \sqcap S_i \not\equiv \bot$	$Q \sqcap S_i \not\equiv \bot$	$Q \sqcap S_i \equiv \bot$	$Q \sqcap S_i \equiv \bot$
Schéma				
Distance	H tel que $H \sqcap S_i \sqsubseteq Q$	H tel que $H \sqcap Q \sqsubseteq S_i$	G tel que $G \sqcap K \equiv S_i$ et $K \sqcap Q \not\equiv \bot$	G tel que $G \sqcap K \equiv Q$ et $K \sqcap S_i \not\equiv \bot$
Ordre	H doit être : -maximal p/r à $\sqsubseteq$ -ou minimal p/r sa taille syntaxique	H doit être : -maximal p/r à $\sqsubseteq$ -ou minimal p/r sa taille syntaxique	G doit être : -maximal p/r à $\sqsubseteq$ -ou minimal p/r sa taille syntaxique	G doit être : -maximal p/r à $\sqsubseteq$ -ou minimal p/r sa taille syntaxique
Cf.	(Colucci S. *et al.*, 2005)			

Tableau 4. : *Raisonnements de découverte 1-1 avec calcul de distance*

Le principe du raisonnement de contraction est similaire : il permet de passer du cas de disjonction au cas d'intersection non vide par négociation. La négociation consiste à identifier dans les services S_i (cas 1), resp. dans la requête Q (cas 2), les contraintes pouvant être abandonnées. Autrement dit, dans le cas 1, il s'agit de découvrir les S_i pouvant s'écrire comme la conjonction de deux descriptions G et K telles que K et Q ont une intersection non vide, G pouvant être négociée (par exemple abandonnée). Le service d'origine est S_i, le service négocié est K. Dans le

cas 2, le principe est le même en échangeant Q et S_i, c'est-à-dire en négociant non plus sur les services mais sur la requête.

Que ce soit pour l'abduction ou la contraction, on remarque que c'est la partie sur laquelle on négocie qui constitue la distance entre Q et les S_i. C'est donc de manière logique qu'on cherche à minimiser la quantité d'informations contenue dans cette partie. C'est ce qui explique que l'ordre proposé soit basé sur une recherche de la plus petite taille syntaxique, ou bien sur une recherche de maximalité par rapport à la subsomption, pour cette partie.

Exemple : En reprenant l'exemple précédent, on voit :

- qu'on peut appliquer un raisonnement d'abduction à (S_4,Q) et (S'_4,Q) pour passer du cas d'intersection à un cas de subsomption ou de plug-in (suivant qu'on ajoute des contraintes aux services ou à Q), et
- qu'on peut appliquer un raisonnement de contraction à (S_5,Q), (S'_5,Q) et (S_6,Q) pour passer du cas de disjonction à un cas d'intersection (suivant qu'on ajoute des contraintes aux services ou à Q).
- Si on ajoute ∀ tv.CRT à Q, alors Q est subsumé par S'_4. La distance calculée par abduction est donc ∀ tv.CRT. Dans ce cas, c'est l'utilisateur qui aura fait une nouvelle offre.
- Si on abandonne dans S_5 la contrainte ≥ 3etoile (on suppose que S_5 est déplié, c'est-à-dire que HebergHautGamme a été remplacé par sa définition), alors S_5 devient équivalent à S_3. Ainsi, dans ce cas, la contraction permet de passer d'un cas de disjonction à un cas de plug-in, et la distance calculée est ≥ 3etoile.
- Si on abandonne ¬Heberg dans S_6, on obtient une nouvelle version de S_6 qui est en intersection avec Q, et plus en disjonction. ¬Heberg est la distance calculée par contraction.

Raisonnements 1-N sans calcul de distance

Le Tableau 5 s'intéresse aux raisonnements d'arité 1-N qui ne calculent aucune distance entre Q et E.

On rappelle que dans ce cas E est une conjonction de S_i. Le raisonnement d'équivalence consiste à chercher parmi toutes les conjonctions possibles de plusieurs S_i celles qui sont équivalentes à Q. Le raisonnement de subsomption maximale permet de découvrir les conjonctions de S_i les plus grandes par rapport à la subsomption qui sont subsumées par Q. Le raisonnement de subsomption maximale est clairement plus flexible que celui d'équivalence, puisque le cas d'équivalence est toujours le cas de subsomption maximale, alors que l'inverse n'est pas vrai. Originellement, ces cas ont été définis dans des travaux concernant le problème de répondre à une requête en utilisant des vues (« query answering using views ») (Halevy A., 2001).

Exemple : Nous reprenons la requête de l'exemple courant de la section 2.4 privée des conjoncts « ∀ lieuArr.≤ 100000hab » et « ∀ aniAccept. ⊥ » :

$$Q_3 \equiv \quad Vol \sqcap \forall lieuArr.Ville \sqcap \forall escale.\forall duree. \leq 1heure \sqcap \\ Heberg \sqcap \leq 2etoile \sqcap \forall equip.Piscine \sqcap \forall tv.EcranPlat$$

On suppose qu'on a l'ontologie de 8 services de la section 2.4

La recherche de conjonctions de services équivalentes ne donne pas de résultat. Cependant il existe 3 ensembles de services dont les conjonctions sont consistantes et maximalement contenues dans Q3. Ce sont {T1, T2, H1}, {T2, T3, H1} et {T3, H1, H2}.

La requête Q, de l'exemple de la section 2.3, n'a que deux conjoncts de plus que Q_3 (« ∀ lieuArr.≤ 100000hab » et « ∀ aniAccept. ⊥ »). Pourtant, elle ne possède aucune réécriture maximalement contenue dans l'ontologie précédente des 8 services. Ceci peut surprendre puisqu' elle n'est pas très différente de Q_3. On se rend dès lors bien compte de l'aspect très contraignant des relations d'équivalence ou d'inclusion maximale dans les découvertes 1-N. A cause de cela, les découvertes basées sur ces deux raisonnements restent assez peu flexibles puisque de nombreuses requêtes resteront sans réponses. Cet état de fait justifie l'existence des découvertes 1-N avec calcul de distance présentées au paragraphe suivant.

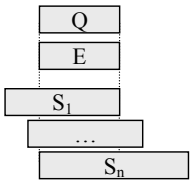
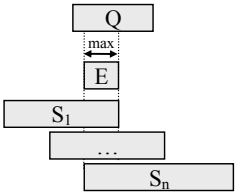
Nom	Equivalence	Subsomption maximale ("maximal containment")
Relation	Q ≡E	E ⊑Q et E ≢ ⊥
Schéma		
Cf.	(Baader F., *et al.*, 2000)	(Halevy A., 2001)

Tableau 5. : *Raisonnements de découverte 1-N sans calcul de distance*

Raisonnements 1-N avec calcul de distance

Le Tableau 6 s'intéresse aux raisonnements d'arité 1-N qui calculent une distance entre Q et E. A notre connaissance, il n'en existe que 2 : les meilleures couvertures d'un concept en utilisant une terminologie, et l'abduction 1-N. On rappelle que dans ces cas E est une conjonction de S_i.

Le raisonnement des meilleures couvertures s'effectue en trois étapes :

– D'abord, on cherche parmi toutes les conjonctions possibles de plusieurs S_i (cf point 1. Tableau 6) celles qui sont consistantes (i.e. non contradictoires) par conjonction avec Q (cf point 2. Tableau 6) et qui ont au moins une information commune avec Q (cf point 3. Tableau 6). Pour assurer ce dernier point, on calcule une première différence entre Q et E, appelée $Rest_E(Q)$, qui contient les informations demandées par l'utilisateur dans Q mais auxquelles E ne répond pas. Pour assurer l'existence d'une information commune, on doit vérifier que $Rest_E(Q)$ est différent de Q. C'est le sens du point 3.

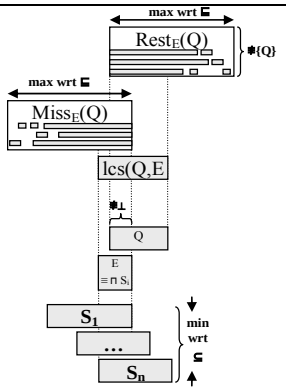
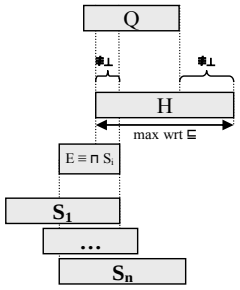
Nom	Meilleure couverture de concept en utilisant une terminologie	Couverture par abduction
Relation	1.E := S1 ⊓…⊓Sn 2.Q ⊓E ≢ ⊥ 3.RestE(Q) ≠{Q} avec RestE(Q) := Q – lcs(Q,E)	1.E := S1 ⊓…⊓Sn 2.Q ⊓E ≢ ⊥ 3.RestE(Q) ≠{Q} avec RestE(Q) := Q – lcs(Q,E)
Schéma		
Distance	La distance est composée de $Rest_E(Q)$ et de $Miss_E(Q)$ définis comme suit : $Rest_E(Q)$:= Q – lcs(Q,E) := {C|C⊓lcs(Q,E)≡Q et C max p/r à ⊑} $Miss_E(Q)$:= E – lcs(Q,E) := {C|C⊓lcs(Q,E)≡E et C max p/r à ⊑}	La distance est une description H telle que : -H ⋢Q -H⊓E ≢ ⊥ -H⊓E ⊑Q
Ordre	$Rest_E(Q)$ doit être maximal p/r à ⊑ ou minimal p/r à sa taille syntaxique. Puis, $Miss_E(Q)$ doit être maximal p/r à ⊑ ou minimal p/r à sa taille syntaxique. Enfin, l'ensemble des S_i qui forment E doit être minimal par rapport à l'inclusion ensembliste.	H doit être maximal p/r à ⊑ ou minimal p/r à sa taille syntaxique.
Cf.	(Rey C., 2004), (Rey C. *et al.*, 2007)	(Di Noia T. *et al.*, 2004)

Tableau 6. : *Raisonnements de découverte 1-N avec calcul de distance*

– Puis, parmi les précédentes conjonctions de S_i, on cherche celles qui ont le plus d'informations en commun avec Q (cf la distance et l'ordre). Pour ce faire, on essaie de minimiser la quantité d'informations présentes dans les deux différences

entre Q et E. D'abord dans le $Rest_E(Q)$, puis dans la deuxième différence, appelée $Miss_E(Q)$, qui contient les informations de E qui n'étaient pas demandées initialement dans Q.

– Enfin, afin d'éviter des conjonctions de services dans lesquelles apparaîtraient plusieurs services équivalents, on ne garde que les conjonctions de services dont l'ensemble des services est minimal par rapport à l'inclusion ensembliste.

Dans ce raisonnement de réécriture, la distance entre Q et E est composée à chaque fois du $Rest_E(Q)$ et du $Miss_E(Q)$ qui sont potentiellement des ensembles de descriptions[21]. Le $Rest_E(Q)$ est la partie de Q incomprise par E ; il peut donc être considéré comme une nouvelle requête en tant que telle que l'on peut transmettre à un autre système de découverte possédant une autre offre de services. En ce sens, le raisonnement des meilleures couvertures s'adapte bien à des architectures pair-à-pair (Baina *et al.,* 2004). De plus il généralise les raisonnements 1-N sans calcul de distance. En effet, l'équivalence et la subsomption sont des cas de meilleures couvertures.

Comme son nom l'indique, le raisonnement de couverture par abduction reprend le principe de l'abduction en le généralisant à des conjonctions de services. Ainsi, il s'agit dans ce raisonnement de trouver des conjonctions E de S_i qui, idéalement, sont subsumées par Q, ou alors pour lesquelles on doit être capable de trouver une description H contenant les contraintes à ajouter pour que la conjonction de E et H soit subsumée par Q. Ainsi, comme dans sa version 1-1, la couverture par abduction permet de passer des cas de conjonctions E d'intersection non vide avec Q, à des cas de conjonctions E subsumées par Q, éventuellement maximalement.

Exemple : Reprenons l'ontologie précédente contenant les 8 services T_1, T_2, T_3, H_1, H_2, H_3, H_4 et V_1, ainsi que la requête Q entière de la section 2.4. On a dit précédemment qu'il n'y avait aucune conjonction de services équivalente ou même maximalement contenue dans Q. Cependant, il en existe beaucoup qui ont une intersection non vide (a priori) avec Q. Par exemple $\{T_1, T_2, H_1\}$, dont la conjonction était maximalement contenue dans Q_3, n'est pas inconsistante avec Q. On est donc un cas d'intersection avec Q (et non de disjonction), avec une arité 1-N. L'abduction 1-N peut donc être utilisée ici pour trouver ce qu'il faut ajouter à $\{T_1, T_2, H_1\}$ pour passer à un cas de subsomption. Comme ce qui différencie Q de Q_3 est le couple de conjoncts (« $\forall$ lieuArr.$\leq$ 100000hab », « $\forall$ aniAccept. $\perp$ »).

A partir de ces 2 conjoncts, on peut obtenir le résultat de l'abduction : il faut ajouter $H = \forall$ lieuArr.$\leq$ 100000hab $\sqcap$ $\forall$ aniAccept.$\geq$11 kg à $T_1 \sqcap T_2 \sqcap H_1$ car on alors un H maximal par rapport à la subsomption tel que :

---

[21] Ttraitement d'éventuelles inconsistances lors du calcul des $Rest_E(Q)$ et du $Miss_E(Q)$

[23] Les deux premiers ensembles sont par ailleurs maximalement contenus dans Q_3. Il est en effet logique que l'on retrouve certaines conjonctions maximalement contenues étant donné que les meilleures couvertures généralisent l'inclusion maximale (i.e. une conjonction maximalement contenue est une meilleure couverture)

$$\underbrace{T1 \sqcap T2 \sqcap H1 \sqcap}_{E} \underbrace{\forall \text{ lieuArr.} \leq 100000\text{hab} \sqcap \forall \text{ aniAccept.} \geq 11 \text{ kg}}_{H} \sqsubseteq Q$$

La distance générée par abduction sera donc la conjonction H suivante ∀ lieuArr.≤ 100000hab ⊓ ∀ aniAccept.≥11 kg. Cette distance est l'expression de ce que le fournisseur aura pu accorder au client durant une phase de négociation. On voit donc l'intérêt de l'abduction après une recherche de conjonctions maximalement contenues infructueuse.

Les meilleures couvertures définissent une relation de découverte plus fine que l'inclusion maximale. Dans l'exemple, Q possède trois meilleures couvertures (déjà données à la section 2.4) : {T3, H1, H2}, {T2, T3, H1} et {T1, T2, H1, H2}, alors qu'il n'existe aucune conjonction maximalement contenue dans Q[23]. L'intérêt des meilleures couvertures est donc d'être une relation plus flexible que l'inclusion maximale et aussi de calculer une notion de distance plus fine que celle obtenue par abduction, puisque composée du $Rest_E(Q)$ et du $Miss_E(Q)$.

Meilleures couvertures	Rest	Miss
$T_2 \sqcap T_3 \sqcap H_1$	{ $\forall lieuArr. \leq 100000hab \sqcap \forall aniAccept \forall transport.\neg Cage \sqcap \forall aniAccept. \geq 1transport$, $\forall lieuArr. \leq 100000hab \sqcap \forall aniAccept.\neg Chien$, $\forall lieuArr. \leq 100000hab \sqcap \forall aniAccept. \geq 11kg$ }	{ $\forall lieuArr.\forall situe.France \sqcap \forall escale.\forall mag.BoutiqueLuxe \sqcap \forall escale.\forall mag. \leq 0DutyFree \sqcap \geq 1escale \sqcap \geq 2etoile \sqcap \forall equip.Tennis \sqcap \forall tv.\forall chaine.Satellite \sqcap \forall tv. \geq 10chaine$ }
$T_3 \sqcap H_1 \sqcap H_2$		{ $\forall lieuArr.\forall situe.France \sqcap \geq 1escale \sqcap \geq 2etoile \sqcap \forall equip.Tennis \sqcap \forall tv.\neg EcranPlat$ }
$T_1 \sqcap T_2 \sqcap H_1 \sqcap H_2$		{ $\forall escale.\forall duree. \geq 2heure \sqcap \forall escale. \geq 1duree \sqcap \geq 2etoile \sqcap \forall equip.Tennis \sqcap \forall tv.\neg EcranPlat$ }

Tableau 7. : *$Rest_E(Q)$ et $Miss_E(Q)$ de chaque meilleure couverture*

Le tableau 7 présente les $Rest_E(Q)$ et $Miss_E(Q)$ de chaque meilleure couverture. On peut voir que les 3 meilleures couvertures ont le même $Rest_E(Q)$ (ce qui n'est pas forcément le cas). Ce $Rest_E(Q)$ est composé de 3 descriptions : chacune d'entre elles représente la partie de la requête à laquelle les meilleures couvertures ne répondent pas. Autrement dit, chaque description du $Rest_E(Q)$ représente ce qu'il faudrait ajouter par conjonction à toute meilleure couverture pour que celle-ci couvre toutes les demandes formulées au travers de la requête. Dès lors, on peut utiliser ce $Rest_E(Q)$ soit pour négocier avec le fournisseur et lui faire modifier son offre, soit pour trouver d'autres services sur d'autres systèmes et d'autres fournisseurs en considérant le $Rest_E(Q)$ comme une nouvelle requête.

Par exemple, si on prend la troisième description du $Rest_E(Q)$ alors on pourra demander au fournisseur s'il peut réaliser la même offre avec une arrivée dans une ville de moins de 100000 habitants et en autorisant les animaux de plus de 11kg seulement.

En regardant les résultats de l'abduction 1-N et des meilleures couvertures, on pourrait croire que ces deux raisonnements calculent les mêmes résultats. On peut montrer qu'en effet ils sont très proches dans la mesure où le H calculé par abduction est souvent égal au $Rest_E(Q)$. Pour certains cas cependant, les meilleures couvertures calculent un $Rest_E(Q)$ portant plus d'informations que la distance H obtenue par abduction.

Requête Q et couverture de services E	Rest et miss obtenus par les meilleures couvertures	Distances obtenues par abduction
$Q \equiv \forall R.(A \sqcap \neg B \sqcap C)$ $E \equiv \forall R.(\neg A \sqcap B \sqcap C)$ (les services composants E ont été remplacés par leur définition)	Partie de Q non prise en compte dans E : $Rest_E(Q) \equiv \forall R.(A \sqcap \neg B)$	Ce que le fournisseur doit ajouter à son offre de services E : $H = \forall R.A$ ou bien $H = \forall R.\neg B$
	Partie de E non demandée dans Q : $Miss_E(Q) \equiv \forall R.(\neg A \sqcap B)$	Ce que le client doit ajouter à sa demande Q : $H = \forall R.\neg A$ ou bien $H = \forall R.B$

Le tableau précédent nous expose un cas où le $Rest_E(Q)$ contient directement toutes les parties de Q auxquelles E ne répond pas, alors que la distance H qu'on obtient par abduction n'en contient qu'une partie. Ce genre de cas reste cependant marginal.

3.4 Vue d'ensemble

La section précédente montre que l'on s'oriente de plus en plus vers des raisonnements dont le but est une plus grande flexibilité. La contrepartie est qu'ils sont plus difficiles à définir, et surtout plus complexes à calculer. C'est ce qui explique qu'ils ne soient étudiés, pour l'instant, que pour des sous-langages de OWL-DL, le standard actuel du web sémantique. Cette remarque vaut particulièrement pour les raisonnements 1-N : dès lors que les solutions sont des conjonctions de services, la complexité des raisonnements semble imposer que ceux-ci soient étudiés pour des langages pas trop expressifs, et notamment pour la logique de description *ALN*.

Indépendamment, les raisonnements de découverte sont déjà très utiles et utilisés. Mais rien n'empêche de les combiner pour obtenir des raisonnements plus fins ou mieux adaptés à une situation donnée. C'est notamment le cas des raisonnements 1-1 sans calcul de distance. Dans (Gonzàlez-Castillo J. *et al*, 2001) (Li L. *et al*, 2003) (Paolucci M. *et al.,* 2002) (Trastour D. *et al.*, 2001) (Trastour D. *et al.*, 2002), les approches de découverte sont essentiellement des ordonnancements des raisonnements 1-1 sans calcul de distance. L'ordonnancement type est le

suivant : on cherche d'abord des services équivalents, puis, s'il n'en existe aucun, des services subsumés ou plug-in, puis des services d'intersection non vide avec la requête, et enfin des services ayant un super concept commun avec la requête. Si tous les cas précédents sont infructueux, on conclut à l'absence de services pertinents pour répondre à la requête. Ces approches n'étant basées que sur la subsumption, elles peuvent être implémentées très efficacement avec les outils de raisonnement existant pour des langages très expressifs comme OWL-DL.

A cette approche d'ordonnancement des raisonnements peut s'ajouter la notion de négociation (Colucci *et al*, 2005). L'utilisation de l'abduction et de la contraction permet par négociations successives de passer d'une situation sans solution (cas de disjonction), à un cas de subsomption, voire d'équivalence. A notre connaissance, cependant, seuls les raisonnements 1-1 sont concernés par ces approches mêlant ordonnancement et négociation. L'existence de la couverture par abduction (raisonnement 1-N) incite à penser que des approches 1-N avec ordonnancement de raisonnements et phases de négociation (par abduction et éventuellement contraction) vont bientôt voir le jour.

En termes de flexibilité, les 2 raisonnements 1-N avec calcul de distance apparaissent comme les meilleurs actuellement, pour une approche purement logique. Les meilleures couvertures offrent une grande flexibilité notamment grâce aux deux calculs de différence ($Rest_E(Q)$ et $Miss_E(Q)$) et au fait que ce raisonnement généralise les deux raisonnements 1-N sans calcul de distance. Cependant, il ne permet pas de traiter les requêtes inconsistantes avec l'offre de services, ce que permettrait une généralisation 1-N du raisonnement de contraction sur le modèle de la généralisation de l'abduction.

Raisonnement	Langage	Complexité
Subsomption 1-1 (Cuenca Gra B. *et al.*, 2006)	OWL-DL	NEXP-Temps complet
Abduction 1-1 (minimalité p/r à la taille syntaxique) (Colucci S. *et al.*, 2004)	$\mathcal{ALN}$	Calcul d'une solution : NP difficile
Subsomption maximale 1-N (Beeri C. *et al.*, 1997)	$\mathcal{ALN}$	Décidable en temps polynomial Calcul d'une solution : exponentiel en temps
Meilleures couvertures 1-N (Rey C., 2004)	$\mathcal{ALN}$	Calcul de toutes les solutions : NP difficile et doublement exponentiel en temps
Couverture par abduction 1-N (minimalité p/r à la taille syntaxique) (Di Noia T. *et al.*, 2004)	$\mathcal{ALN}$	NP difficile

Tableau 8. : *Résultats de complexité de raisonnements de découverte*

La philosophie des meilleures couvertures étant l'automatisation de la découverte, la négociation n'a pas été prévue en tant que telle dans le déroulement de la découverte. Cependant, comme on l'a vu dans les exemples, les $Rest_E(Q)$ et

$Miss_E(Q)$ peuvent être utilisés a posteriori de la découverte pour affiner la mise en correspondance. En permettant l'ajout ou l'abandon de contraintes, la négociation est le but premier de l'abduction et de la contraction. Nous conclurons cette comparaison en remarquant que le calcul d'abduction est une généralisation du calcul de différence utilisé dans les meilleures couvertures, ce qui explique en partie la proximité des deux raisonnements. De plus, les deux ont été étudiés pour la logique de description *ALN*. Meilleures couvertures et abduction montrent clairement le chemin à suivre dans le domaine des raisonnements pour la découverte de services abstraits.

Le Tableau 8 récapitule quelques résultats de complexité pour les raisonnements évoqués, avec les langages correspondants.

4. De la découverte à la composition de services web

Dans cette première partie, nous avons donné un panorama détaillé des différents éléments caractérisant des mécanismes de découverte de services web sémantiques. Nous avons proposé une classification des raisonnements de découverte de services web sémantiques décrits abstraitement. Cette classification met en exergue les principaux facteurs qui permettent une plus grande flexibilité : (i) les découvertes 1-N (associant à une requête un ensemble de services), (ii) la définition et le calcul de distances sémantiques entre concepts (pour une meilleure caractérisation et un meilleur ordonnancement des services découverts), et (iii) la composition de raisonnements 1-N selon un processus de négociation.

L'exemple de la section 2.4 nous pousse à constater que la découverte de services abstraits, si elle est nécessaire, n'est pas suffisante pour automatiser le processus complet de traitement des services, de la requête utilisateur à l'exécution dynamique des services distants. Ainsi entrent en jeu les problématiques de la découverte des services décrits fonctionnellement et de celle des services décrits par leur comportement. De part la nature des descriptions fonctionnelles et comportementales, les découvertes associées peuvent être vues comme des premières approches de composition de services.

Prendre en compte les résultats de la découverte de services abstraits dans les découvertes fonctionnelles et comportementales, et donc dans la composition, est un enjeu majeur pour atteindre un traitement automatisé de bout en bout. De manière générale, on doit considérer les problèmes de découverte et de composition comme étroitement corrélés. De plus, pour des raisons d'efficacité (les problèmes ayant souvent une combinatoire exponentielle) la résolution rapide du problème de composition nécessite d'explorer un ensemble réduit de services. Cette réduction est effectuée par la découverte. En général, tous les modèles de composition partent d'un ensemble de services préalablement découvert.

Il est par exemple possible de généraliser certains raisonnements de découverte de services abstraits à des descriptions fonctionnelles de services. Par exemple, (Benatallah *et al*, 2003) proposent une application des meilleures couvertures aux

descriptions fonctionnelles de services exprimés en termes d'entrées et sorties avec OWL-S. Ceci laisse à penser que ces raisonnements peuvent fournir une première approche de composition fonctionnelle par itérations successives de découvertes fonctionnelles. C'est le même principe qui est d'ailleurs à la base de certaines méthodes de composition fonctionnelle par des techniques de chaînage arrière : après avoir découverts des services qui fournissent les sorties d'une requête, on réitère la découverte pour trouver d'autres services qui fournissent en sorties les entrées des premiers services découverts (Lécué *et al*, 2006). D'autres travaux se situent à mi-chemin entre la découverte et la composition fonctionnelle. Par exemple, (Baader *et al*, 2005) proposent un formalisme centré sur la notion d'action et basée sur une logique de description pour modéliser et raisonner sur les pré-conditions et les effets de l'exécution d'un service sur l'état du monde. Les deux raisonnements qu'ils définissent sont l'exécution et la projection qui assurent respectivement que les pré-conditions d'un service sont satisfaites et les effets atteints. Ils peuvent aussi bien servir dans la découverte d'un service, en permettant de vérifier que ses pré-conditions sont toutes satisfaites, que dans la découverte d'une succession de services en assurant que les effets recherchés sont bien atteints après l'exécution de tous les services. Ce dernier cas est déjà un cas de composition.

Dans la deuxième partie de l'article, nous aborderons donc la composition de services web sémantiques. Nous présenterons les principales approches. Nous les illustrerons par des exemples empruntés à l'industrie. Nous dresserons un panorama rapide mais précis sur les technologies et architectures majeures en compétition. Puis nous terminerons sur une vision prospective de verrous technologiques et de recherche qui freinent la généralisation de la technologie des services web sémantiques dans l'industrie.

5. Bibliographie

Alonso G., Casati F., Kuno H. And Machiraju V., Web services: Concepts, Architectures and Applications. Springer-Verlag, 2004.

Baader F., Küsters R., and Molitor R., "Rewriting concepts using terminologies", In Proc. Of KR2000, p. 297-308, 2000.

Baader F. , Calvanese D., McGuinness D., Nardi D., Patel-Schneider P., The Description Logic Handbook. Theory, Implementation and Applications Edited by Cambridge University Press, 2003.

Baader F., Lutz C., Milicic M., U. Sattler U. and Wolter F., A Description Logic Based Approach to Reasoning about Web Services. In Proceedings of the WWW 2005 Workshop on Web Service Semantics (WSS2005), Chiba City, Japan, 2005

Baina K., Benatallah B., Paik H-Y., Toumani F, Rey C., Rutkowska A., Harianto B., "WS-CatalogNet: An Infrastructure for Creating, Peering, and Querying e-Catalog Communities". VLDB 2004, 29 August - 3 September 2004 Toronto, Canada, 2004.

Beeri C., Levy A.Y., Rousset M-C., “Rewriting Queries Using Views in Description Logics”. Proceedings of the Sixteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, May 12-14, 1997, Tucson, Arizona. ACM Press 1997.

Benatallah B., Hacid M., Rey C., Toumani F., "Request Rewriting-based Web Service Discovery, The Semantic Web - ISWC 2003, LNCS 2870, pp. 242 – 257 - 2003

Colucci S., Di Noia T., Di Sciascio E., Donini F.M. , Mongiello M., “A Uniform Tableaux-Based Method for Concept Abduction and Contraction” in Description Logics. Proceedings of ECAI 2004, pp 975-976 – 2004.

Colucci S., Di Noia T., Di Sciascio E., Donini F.M., and Mongiello M.. “Concept abduction and contraction for semantic-based discovery of matches and negotiation spaces in an e-marketplace ». Electronic Commerce Research and Applications, 4(4), p. 345–361, 2005.

Cuenca Gra B., OWL 1.1 Web Ontology Language Tractable Fragments W3C Member Submission 19 December 2006, http://www.w3.org/Submission/owl11-tractable/

Di Noia T., Di Sciascio E., Donini F.M., “Extending and computing the concept covering for the semantic web”. Research Report, Politecnico di Bari- Dipartimento di Elettrotecnica ed Elettronica, Number 21/04/S – 2004.

Halevy A.. Answering Queries Using Views: A Survey. VLDB Journal, 10(4):270–294, 2001. http://www.cs.washington.edu/homes/alon/site/files/view-survey.ps

Kaufer F. and Klusch M., "WSMO-MX: A Logic Programming Based Hybrid Service Matchmaker", IEEE ECOWS 2006

Keller U., Lara R., Lausen H., Fensel D., "Semantic Web Service Discovery in the WSMO Framework", Chapter 12 of Semantic Web Services: Theory, Tools and Applications, Cardoso, Idea Group Publishing, 2007

Klusch M. and Fries B, "Hybrid OWL-S Service Retrieval with OWLS-MX: Benefits and Pitfalls", 1st International Workshop on Service Matchmaking and Resources Retireval in the Semantic Web, ISWC 2007

Lécué F. and Léger. A., “A formal model for semantic web service composition”. In ISWC the 5th International Semantic Web Conference, November 2006.

Li L. and Horrocks I., “A software framework for matchmaking based on semantic web technology”. In Proceedings of the Twelfth International World Wide Web Conference, WWW'2003, 2003.

McIlraith S., Son Cao Tran, Zeng H., "Semantic Web Services", IEEE Intelligent Systems, pp. 46-53, March-April 2001

Paolucci M., Kawamura T., Payne T.R., and Sycara K., “Semantic matching of web services capabilities ». Proceedings of the First International Semantic Web Conference, LNCS 2342, p. 333–347. Springer-Verlag, 2002.

Rey C., Benatallah B., Léger A. and Toumani F., “Covering concepts using terminologies in the ALN language”. Technical report, LIMOS, Clemont-Ferrand, France, http://www.isima.fr/rey/alnBcov.pdf, 2007.

Rey C., Découverte des meilleures couvertures d'un concepts en utilisant une terminologie. Application à la découverte de services web sémantiques PhD Thesis Université Blaise Pascal Clermont 2 France, 2004.

Sivashanmugam, K.; Verma, K.; Sheth, A.; and Miller, J.. "Adding semantics to web services standards". In ICWS 2003 proceedings , p. 395–401.

Tim Berners-Lee, James Hendler, and Oral Lassila. The semantic web. Scientific American, 284(5):34-43, May 2001

Trastour D., Bartolini C., and Gonzalez-Castillo J., "A semantic web approach to service description for matchmaking of services". In Proceedings of the International Semantic Web Working Symposium, SWWS, 2001.

Trastour D., Bartolini C., and Preist C., "Semantic web services: Semantic web support for the business-to-business e-commerce lifecycle". In Proceedings of the eleventh international conference on World Wide Web, pages 89–98. ACM Press, 2002