Les langages du web sémantique

Jean-François Baget[#], Étienne Canaud[*], Jérôme Euzenat[#], Mohand Saïd-Hacid[*]
[#] INRIA Rhône-Alpes [*] LISI

1 Présentation et importance de la problématique du point de vue des usages

Le web sémantique doit pouvoir être manipulé par les machines. Dans l'état actuel de la technologie, il est alors nécessaire de disposer de langages pour :

- exprimer les données et les métadonnées (cf. #PartieAnnot#);
- exprimer les ontologies (cf. #PartieOnt#);
- décrire les services (cf. #PartieWS#).

Certes, il existe déjà des langages développés pour ces activités indépendamment du web sémantique (KIF ou WPDL). Ils ne sont pas utilisés tels quels dans le web sémantique car il est nécessaire de leur permettre d'accepter les caractères propres au web à savoir sa distribution (il faut être capable de tirer parti d'information dont on ne dispose pas localement) et son ouverture (n'importe qui peut ajouter de l'information à tout instant).

Disposer de chacun de ces langages est indispensable au développement des fonctionnalités correspondantes du web sémantique. Ces langages permettrons diverses applications nouvelles telles que :

- La recherche d'information fondée sur des descriptions formelles ;
- La composition de services en fonction de leurs descriptions ;
- L'interconnexion de catalogues sur la base de leur description.

Le but du web sémantique est principalement que les services soient mieux rendus sans engendrer de surcharge pour les utilisateurs. Dans cette perspective, les usages ne devraient se voir impacter que positivement par les langages développés. Mais l'idée du web est que les usagers en soient les contributeurs. C'est en ce sens que les langages développés pour le web sémantique pourront avoir un impact sur ceux qui les utiliseront pour décrire leurs ressources voire leurs services.

Par ailleurs, même si ces langages étaient destinés à rester cachés (c'est-à-dire accessible au travers d'une application plus conviviale), les langages utilisés risquent de leur imposer indirectement leurs contraintes. Ainsi, dans une application de recherche d'information, l'expressivité du langage de requêtes contraindra la forme et l'étendue de la réponse. Ces problèmes seront illustrés dans les sections 3 et 4.

2 Méthodes, techniques, outils existants sur lesquels on peut s'appuyer

XML est le langage de base. Il a l'avantage d'être fait pour la communication en réseau et de disposer de nombreux outils. Il est donc naturellement utilisé pour encoder les langages du web sémantique. Mais il a surtout la propriété d'être un métalangage (une description de type de document, DTD, permet de décrire la grammaire des documents admissibles).
Bien entendu, ceci ne permet pas à une machine de manipuler sémantiquement un document. Mais cela a la vertu de permettre une manipulation syntaxique de tous les documents. Ainsi, une annotation sera attachée de la même manière à un paragraphe, un exposant dans une formule mathématique ou un polygone dans un dessin parce que

ceux-ci sont encodés en XML. C'est cette propriété qui permet d'insérer des éléments du Dublin-core dans une ontologie et d'annoter des documents par de la connaissance formalisée.
Cette compatibilité entre les langages décrits en XML permet de construire les langages présentés ci-dessous et de les considérer comme des documents XML.
Mais XML est limité car il ne dispose pas d'une sémantique (au sens logique de sémantique dénotationnelle d'un langage). Rien ne justifie donc les raisonnements ou manipulations appliquées à des documents XML. C'est pourquoi il est nécessaire de développer d'autres langages.

La seconde source d'inspiration est celle de la représentation de connaissance et notament les langages de représentation de connaissance que sont les logiques de descriptions et les réseaux sémantiques (que nous considérerons ici sous leur aspect plus avancé des graphes conceptuels). Ces langages permettent d'exprimer la connaissance de nature ontologique (décrire des classes d'entités, les relier par spécialisation, décrire et typer leurs attributs) ou assertionnelle (décrire l'état du monde par des individus en relations entre eux, individus et relations étant décrits dans l'ontologie).
Depuis une quinzaine d'années ces langages sont définis par leur sémantique et caractérisés par leur décidabilité et complexité. Ceci permet de développer des moteurs d'inférence dont on connaît clairement les limites d'application.
Ces langages ont certaines limitations dans la prise en compte de la nécessaire ouverture du Web (relations entre objets distribués, ajout de connaissance incontrôlé); ils sont donc reconsidérés dans ce contexte.

Enfin, dans le cadre des descriptions de services, les dernières sources d'inspiration sont les langages de description de plans et en particulier les langages de description de "Workflow" permettant d'exprimer de manière abstraite des activités (ou tâches) et leurs dépendances (séquence, parallélisme, synchronisation…). Ces langages sont exécutables par des logiciels qui contrôlent l'exécution du plan à l'aide d'événements prédéfinis (envoi d'un mail, remplissage d'un formulaire, signature d'un engagement…). Le langage le plus emblématique est certainement WPDL, proposé par la "Workflow Management Coalition", parce qu'il est compris par différents logiciels de workflow. Il a été récemment décliné en XML (XPDL).
Ces langages sont en général destinés à être supervisés par les humains qui exécutent les tâches du workflow, ils doivent donc acquérir plus de rigueur dans la description des tâches pour pouvoir être manipulés par des machines dans le cadre du web sémantique.

3 Travaux et résultats existants du web sémantique

Il semble clair que le web sémantique ne pourra voir le jour sans un minimum de standardisation. Différents consortiums et organismes mettent donc les acteurs autour d'une table pour définir les langages à utiliser dans le web sémantique. L'intérêt de cette approche standardisante est bien sûr d'assurer des traitements uniformes sur l'ensemble des documents écrits dans ces langages. Un inconvénient peut être le gel d'autres travaux de recherche plus originaux. Une approche moins contraignante aurait pu être la définition de *métalangages* en permettant de décrire des langages (que ce soit par leur sémantique, en décrivant leurs modèles, ou de façon opérationnelle, en fournissant des règles encodant les mécanismes de raisonnement), cette standardisation n'aurait pas gêné l'ouverture vers d'autres travaux.

Les travaux de standardisation sont aujourd'hui bien avancés : RDF et SOAP sont des recommandations du W3C, TopicMaps une norme ISO, et OWL est sur les pas de RDF.

Nous décrirons ici trois sortes de langages :
- des langages d'assertions (RDF et cartes topiques) ;
- un langage de définition d'ontologies pour le web (OWL) ;
- différents langages de description et de composition de services (UDDI et autres).

Dans les deux premiers cas nous nous appuierons principalement sur les langages proposés par le W3C qui a réussi à faire interagir un grand nombre d'acteurs tant académiques qu'industriels (et à bâtir sur différents langages proposés antérieurement). Ces langages sont munis d'une sémantique formelle, en théorie des modèles. Un des intérêts de munir les langages d'une sémantique formelle est de pouvoir définir de façon naturelle la notion de conséquence : un document RDF est conséquence d'un autre veut dire que toute information contenue dans ce dernier est aussi contenue dans le premier ; et une classe OWL est conséquence d'une autre veut dire que toutes les instances de la seconde sont des instances de la première. Ceci nous permet de comparer des faits (dans RDF) ou des classes (dans OWL), et donc permet d'interroger une base de documents : l'utilisateur peut par exemple définir un document RDF (la question), et lancer un mécanisme de recherche sur le web pour les documents RDF dont la question est une conséquence. Ce seront les réponses à cette question. Ceci ne fait cependant pas de RDF un vrai langage de requêtes. Bien qu'il puisse permette de répondre à certaines questions (y a-t-il un train de Grenoble à Paris partant entre 8h00 et 9h30 demain ?), il ne permet pas d'agir (dans ce cas, réserver les billets). Il faudrait pour cela encapsuler ces langages dans des langages de requêtes similaires à ceux que l'on peut trouve en bases de données, à moins que cette tâche ne soit totalement dévolue aux services.

3.1 Langages d'assertions et d'annotations

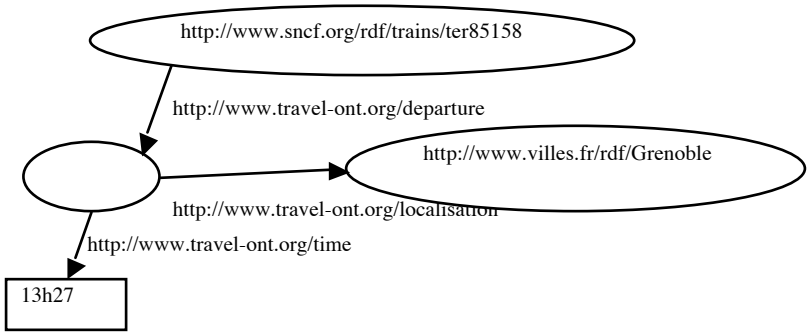
Les assertions affirment l'existence de relations entre des objets. Elles sont donc adaptées à l'expression des annotations que l'on veut associer aux ressources du web. On évoquera principalement RDF ici car il nous semble présenter des avantages déterminants pour la manipulation informatique, mais le formalisme des cartes topiques mérite que l'on s'y intéresse.

3.1.1 RDF

RDF est un langage formel qui permet d'affirmer des relations entre des « ressources ». Il sera utilisé pour annoter des documents écrits dans des langages non structurés, ou comme une interface pour des documents écrits dans des langages ayant une sémantique équivalente (des bases de données, par exemple). RDF est muni d'une syntaxe, et d'une sémantique. Aucun mécanisme d'inférence n'est cependant proposé dans la recommandation.

Un document RDF est un ensemble de triplets de la forme *<sujet, prédicat, objet>*. Ce document sera codé en machine par un document RDF/XML ou N3, mais est souvent représenté sous une forme graphique.
Les éléments de ces triplets peuvent être des URIs (Universal Resource Identifiers), des littéraux ou des variables. Un tel ensemble peut être représenté de façon naturelle par un graphe (plus précisément un multi-graphe orienté étiqueté), où les éléments

apparaissant comme sujet ou objet sont les sommets, et chaque triplet est représenté par un arc dont l'origine est son sujet et la destination son objet (voir figure).



Le dessin précédent illustre une partie d'un tel document RDF (il s'agit d'un exemple fictif, montrant comment la SNCF pourrait donner une interface RDF à sa base de données de voyages). Les ressources de la forme http://... sont des URIs qui identifient des termes définis de façon unique. Notons dans les URIs que certaines ressources sont spécifiques à la SNCF (le train), et que d'autres (departure…) sont issus d'une ontologie dédiée aux voyages. Les objets d'un triplet qui sont des littéraux sont représentés dans un rectangle (ici, 13h27). Le sommet non étiqueté représente une variable. Intuitivement, ce graphe peut se comprendre comme «□le train TER 85158 part de Grenoble à 13h27□. Cette sémantique «□ntuitive□ ne suffisant pas à un traitement automatique, il faut munir les documents RDF d'une sémantique formelle.

La sémantique d'un document RDF est exprimée en théorie des modèles. L'objectif est de donner des contraintes sur les mondes qui peuvent être décrits par un document RDF. L'utilisation de la théorie des ensembles pour décrire ces modèles a deux intérêts□la généricité de la notion d'ensemble (fondement des mathématiques□) et son universalité (culture commune pour ceux qui vont s'intéresser à cette sémantique).
Un document RDF peut aussi être traduit en une formule de la logique positive (sans négation), conjonctive, existentielle du premier ordre (sans symboles fonctionnels), dont les modèles sont identiques à ceux définis par la sémantique directe en théorie des modèles. À chaque triplet <*s, p, o*> on associe la formule atomique *p(o, s)*, où *p* est un nom de prédicat, et *o* et *s* sont des constantes si ces éléments sont des URIs ou des littéraux dans le triplet, et des variables sinon. Le document RDF se traduit par une formule qui est la fermeture existentielle de la conjonction des formules atomiques associées à ses triplets. Ainsi, le document RDF utilisé précédemment en exemple se traduit par la formule□

$$\exists x\ (departure(ter85158, x) \wedge time(x, 13h27) \wedge localisation(x, Grenoble))$$

L'information contenue dans un document RDF R_1 est déjà présente dans le document RDF R_2 si et seulement si la formule logique associée à R_1 est conséquence de celle associée à R_2. Cette «□traduction logique□ de RDF permet de l'identifier à de nombreux autres paradigmes de raisonnement□ la logique, bien sûr, mais aussi les bases de données (Datalog positif) ou les graphes conceptuels.

Bien qu'un mécanisme d'inférence adéquat et complet par rapport à la sémantique (on ne trouve que des conséquences, et toutes les conséquences) soit évoqué dans les propositions du W3C, ceci n'entre pas dans la standardisation. L'objectif est de laisser

la plus grande liberté à ceux qui vont implémenter des outils fondés sur RDF, en n'établissant pas la certification sur les mécanismes de raisonnements. Le rapprochement avec les graphes conceptuels simples permet cependant de préciser ce mécanisme de raisonnement. Il s'agit d'un homomorphisme de graphes étiquetés, pour lequel des algorithmes efficaces (bien qu'il s'agisse d'un problème NP-complet) ont été développés.
RDF propose aussi certains mots-clés réservés, qui permettent de donner une sémantique particulière à des ressources. Ainsi, on peut représenter des ensembles d'objets (*rdf:bag*), des listes (*rdf:sequence*), des relations d'arité quelconque (*rdf:value*)… Ce ne sont cependant pas de réelles extensions du langage présenté ci-dessus, puisqu'une transformation (la réification) permet d'exprimer cette «□sémantique étendue□» dans le langage de base□: R_1 est une conséquence (sémantique étendue) de R_2 si et seulement si *réif*(R_1) est une conséquence (au sens précédent) de *réif*(R_2).

RDFS (pour RDF Schéma) a pour but d'étendre le langage en décrivant plus précisément les ressources utilisées pour étiqueter les graphes. Pour cela, il fournit un mécanisme permettant de spécifier les classes C dont les ressources étiquetées par C seront des instances, comme les propriétés. RDFS s'écrit toujours à l'aide de triplets RDF, en définissant la sémantique de nouveaux mots-clés comme:

- *<ex:Vehicule rdf:type rdfs:Class>* la ressource *ex:Vehicule* a pour type *rdfs:Class*, et est donc une classe□;
- *<sncf:TER8153 rdf:type ex:Vehicule>* la ressource *sncf:TER8153* est une instance de la classe *ex:Vehicule* que nous avons définie□;
- *<sncf:Train rdfs:subClassOf ex:Vehicule>* la classe *sncf:Train* est une sous-classe de *ex:Vehicule*, toutes les instances de *sncf:Train* sont donc des instances de *ex:Vehicule*□;
- *<ex:localisation rdf:type rdfs:Property>* affirme que *<ex:localisation>* est une propriété (une ressource utilisable pour étiqueter les arcs)□;
- *<ex:localisation rdfs:range ex:Ville>* affirme que toute ressource utilisée comme extrémité d'un arc étiqueté par *ex:localisation* sera une instance de la classe *ex:Ville*.

Ce besoin de spécifier davantage les classes est à l'origine du langage dédié aux définitions de classes□OWL.

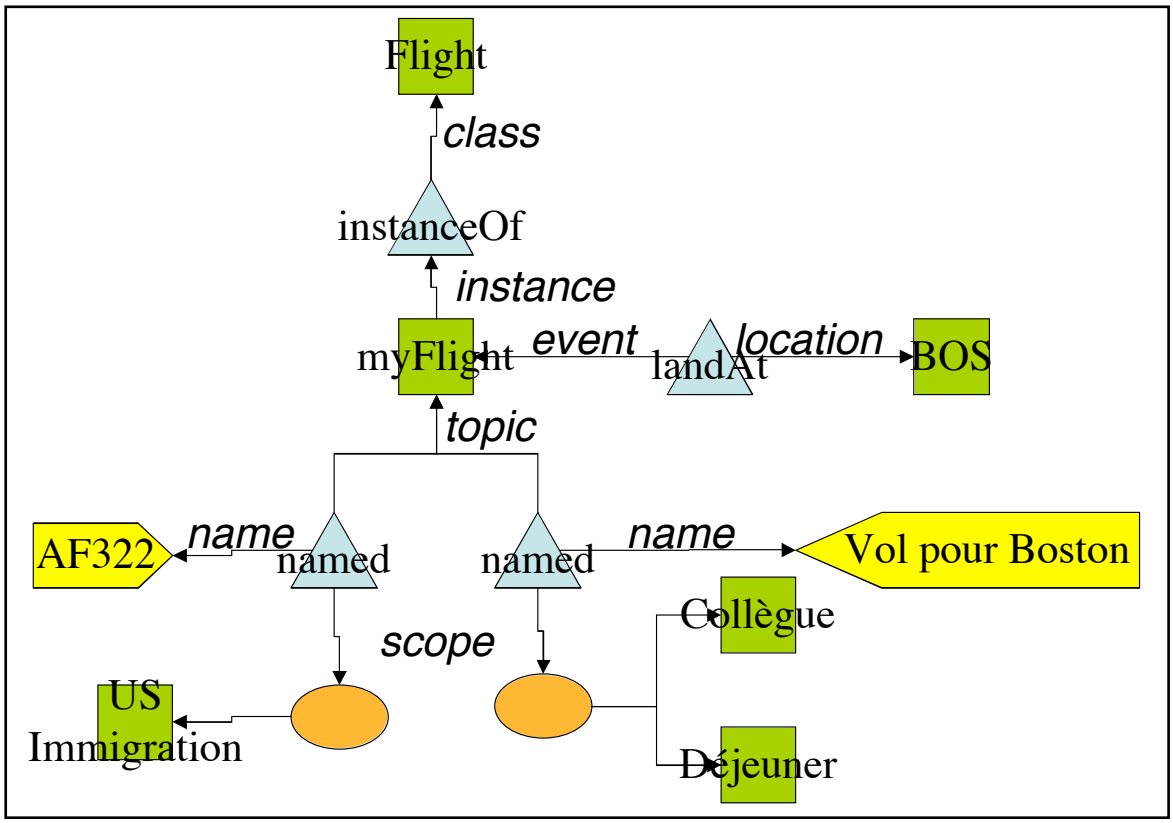
3.1.2 Cartes topiques

Les cartes topiques ("Topic maps") sont un standard ISO issu de HyTime dont le but était d'annoter les documents multimédia. Issu de SGML, il s'est vu récemment attribuer une syntaxe XML (XTM). Par ailleurs, un groupe de l'ISO s'occupe de définir un langage de requêtes pour les cartes topiques (TMQL).
Les cartes topiques sont bâties autour de quatre notions primitives (nous faisons ici abstraction des sujets)□:

- Les "topics" que l'on peut comprendre comme des individus des langages de représentation de connaissances.
- Les noms donnés aux topics: l'une des originalités des cartes topiques est la séparation des concepts et de leurs noms. Cela permet d'avoir plusieurs noms pour le même concept (et donc d'avoir des cartes topiques multilingues) et des noms partagés par plusieurs concepts.

- Les occurrences sont des “proxis” d’entités externes qui peuvent ainsi être indexés par les topics (où les entités littérales lorsque celles-ci sont représentables).
- Les portées, qui sont parfois vues comme une quatrième dimension, permettent de spécifer le contexte dans lequel une relation est valide.

Par exemple, le topic de vol est instancié par myFlight, il a pour nom «□ol pour Boston□ dont la portée est celle de mes discussions au déjeuner avec les colègues et “flight AF322” lors de discussions avec l’immigration américaine.



Si ces quatre dimensions sont spécifiées de manière indépendante, elles ont en réalité interdépendantes□les topics et les noms ont des portées, les topics ont des noms, les portées sont des ensembles de topics…).

Dans la nouvelle syntaxe des cartes topiques, celles-ci sont représentées par des graphes comprenant 3 types de nœuds (topic, association, portée) et un certain nombre de types d’arcs (instance, occurrence, portée, nom). Les relations sont représentées par des nœuds dont les arcs sortant portent des étiquettes identifiant leur rôle. Par ailleurs, différentes interprétations sont données à ces primitives suivant les étiquettes placées sur les arcs et les nœuds. Autant dire que les cartes topiques ne disposent pas d’une sémantique claire et que, au contraire, ses concepteurs ont tendance à considérer que la richesse du langage tient dans les interprétations multiples que l’on peut en faire.
Ceci ne le rend pas un candidat très souhaitable pour le web sémantique malgré ses qualités indéniables. Il existe cependant des outils permettant de tirer parti de manière utile des cartes topiques qui sont utilisés dans un certain nombre d’applications.

3.2 Langages de définitions d’ontologies

RDF, langage dédié à l’expression d’assertions sur les relations entre objets, s’est heurté à la nécessité de définir les propriétés des classes dont ces objets sont instances. Cependant, l’extension à RDFS ne fournit que des mécanismes très basiques pour

spécifier ces classes. Le langage OWL, quant à lui, est dédié aux définitions de classes et de types de propriétés, et donc à la définition d'ontologies. Inspiré des logiques de descriptions (et sucesseur de DAML+OIL), il fournit un grand nombre de constructeurs permettant d'exprimer de façon très fine les propriétés des classes définies. La rançon de cette expressivité est l'indécidabilité du langage obtenu en considérant l'ensemble de ces constructeurs. C'est pour cela que OWL a été fractionné en trois langages distincts :

- OWL LITE ne contient qu'un sous-ensemble réduit des constructeurs disponibles, mais son utilisation assure que la comparaison de types pourra être calculée (un problème de NP, donc « simple » en représentation de connaissances) ;
- OWL DL contient l'ensemble des constructeurs, mais avec des contraintes particulières sur leur utilisation qui assurent la décidabilité de la comparaison de types. Par contre, la grande complexité de ce langage (un de ses fragments est P-SPACE-complet) semble rendre nécessaire une approche heuristique ;
- OWL FULL, sans aucune contrainte, pour lequel le problème de comparaison de types est vraisemblablement indécidable.

La syntaxe d'un document OWL est donnée par celle des différents constructeurs utilisés dans ce document. Elle est le plus souvent donnée sous la forme de triplets RDF. À chaque constructeur est associée une sémantique, en théorie des modèles. Elle est directement issue des logiques de descriptions. La sémantique associée aux mots-clés de OWL est plus précise que celle associée au document RDF représentant une ontologie OWL (elle permet plus de déductions).

Nous donnons ici l'ensemble des constructeurs utilisés dans OWL, dans une syntaxe simplifiée (les mots-clés réservés de OWL, habituellement préfixés de OWL : sont soulignés), ainsi que leur « sémantique intuitive ». Les constructeurs de OWL LITE sont cités les premiers.

OWL LITE

- Reprend tous les constructeurs de RDF (c'est-à-dire fournit des mécanismes permettant de définir un individu comme instance d'une classe, et de mettre des individus en relation),
- Utilise les mots-clés de RDFS (*rdfs:subClassOf, rdfs:Property, rdfs:subPropertyOf, rdfs:range, rdfs:domain*), avec la même sémantique,
- Permet de définir une nouvelle classe (*owl:Class*) comme étant plus spécifique ou équivalente à une intersection d'autres classes,
- *owl:sameIndividualAs* et *owl:differentIndividualFrom* permettent d'affirmer que deux individus sont égaux ou différents,
- Des mots-clés permettent d'exprimer les caractéristiques des propriétés : *owl:inverseOf* sert à affirmer qu'une propriété *p* est l'inverse de *p'* (dans ce cas, le triplet <*s p o*> a pour conséquence <*o p' s*>) ; d'autres caractéristiques sont par exemple la transitivité (*owl:TransitiveProperty*), la symétrie (*owl:SymmetricProperty*),
- *owl:allValuesFrom* associe une classe *C* à une propriété *P*. Ceci définit la classe des objets *x* tels que si <*x P y*> est une relation, alors la classe de *y* est *C* (quantification universelle de rôle en logique de descriptions). *owl:someValuesFrom* encode la quantification existentielle de rôle,
- *owl:minCardinality* (resp. *owl:maxCardinality*) associe une classe *C*, une propriété *P*, et un nombre entier *n*. Ceci définit la classe des objets *x* tels qu'il existe au moins (resp. au plus) *n* instances différentes *y* de *C* avec <*x P y*>. Pour

des raisons d'efficacité algorithmique, OWL LITE ne permet d'utiliser que des entiers égaux à 0 ou 1. Cette restriction est levée dans OWL DL.

OWL DL

- Reprend tous les constructeurs d'OWL LITE,
- Permet tout entier positif dans les contraintes de cardinalité,
- *owl:oneOf* permet de décrire une classe en extension par la liste de ses instances,
- *owl:hasValue* affirme qu'une propriété doit avoir comme objet un certain individu,
- *owl:disjointWith* permet d'affirmer que deux classes n'ont aucune instance commune,
- *owl:unionOf* et *owl:complementOf* permettent de définir une classe comme l'union de deux classes, ou le complémentaire d'une autre classe.

OWL FULL

- reprend tous les constructeurs d'OWL LITE,
- reprend tout RDF Schema,
- permet d'utiliser une classe en position d'individu dans les constructeurs.

Nous n'avons pas cité ici certains constructeurs, qui peuvent être trivialement implémentés grâce à ceux que nous avons évoqués (par exemple *owl:sameClassAs,* servant à affirmer que deux classes sont identiques, peut être écrit grâce à deux *rdfs:subClassOf*). Il serait intéressant d'identifier quels sont les constructeurs primitifs nécessaires pour ces langages, et ceux qui ne sont que des macros.

Des moteurs d'inférence ont déjà été implémentés pour des sous-ensembles significatifs de OWL DL (dans le cadre des logiques de descriptions) et peuvent être utilisés dans divers outils (OilEd, Protégé…).

3.3 Langages de description et de composition de services

Cette partie a pour but de décrire différents langages, architectures et standards concernant les services sur le web (ou web services, cf. #PartieWS#). De nouveaux langages dédiés aux services web sont régulièrement proposés par les organismes de recherche industriels et universitaires. Il ne faut pas perdre de vue que la plupart des langages présentés sont complémentaires et ne répondent pas aux mêmes besoins. Nous allons donc présenter les objectifs et les fonctionnalités des principaux langages consacrés aux services sur le web.

3.3.1 UDDI

Le protocole UDDI (Universal Description, Discovery and Integration) est une plateforme destinée à stocker les descriptions des services web disponibles, à la manière d'un annuaire de style «□Pages Jaunes□. Des recherches sur les services peuvent être effectuées à l'aide d'un système de mots-clés fournis par les organismes proposant les services. UDDI propose également un système de «□Pages Blanches□ (adresses, numéros de téléphone, identifiants…) permettant d'obtenir les coordonnées de ces organismes. Un troisième service, les «□Pages Vertes□, permet d'obtenir des informations techniques détaillées à propos des services et permettent de décrire comment interagir avec les services en pointant par la suite vers un PIP RosettaNet ou une "service interface" WSDL. Le vocabulaire utilisé pour les descriptions obéit à une taxonomie bien précise afin de permettre une meilleure catégorisation des services et des organismes.

De par sa simplicité, UDDI permet de stocker l'ensemble des services web sur un seul serveur, dont le contenu est dupliqué et synchronisé sur plusieurs sites miroirs. Des

implémentations d'UDDI ont été réalisées, et on peut d'ores et déjà enregistrer son entreprise et les services proposés sur UDDI. Cependant, on peut s'interroger sur la réelle efficacité en matière de recherche d'une architecture aussi simple où la sémantique des données est inexistante et où la description des services se limite à des mots-clés sur lesquels aucune approximation n'est possible. De plus, il n'est pas certain que des serveurs uniques puissent supporter la charge du nombre de services à venir.

3.3.2 E-Speak

Bien que son développement ait été arrêté il y a environ un an, E-speak, développé par Hewlett-Packard, a été un des pionniers en matière de service web, avec des débuts qui remontent à 1995. Les buts d'E-speak étaient sensiblement les mêmes que ceux d'UDDI, à savoir la publication et la découverte de services, mais E-speak avait une approche plus générale, en gérant par exemple les services d'impression dans un réseau local d'entreprise.
Les services y sont décrits comme un ensemble d'attributs issus de différents vocabulaires distincts, homogénéisant ainsi la sémantique (tout en ne permettant pas une grande flexibilité). Les vocabulaires sont issus de mêmes groupes de services logiques. Chaque service possède des attributs fixés (nom, description, version…), mais ces attributs n'ont pas de sémantique en eux-mêmes.
Hewlett-Packard a décidé d'incorporer E-speak à UDDI, et il y a fort à parier que ses spécificités devraient bientôt y apparaître.

3.3.3 WSDL

WSDL est un langage basé sur XML servant à décrire les interfaces des services web, c'est-à-dire en représentant de manière abstraite les opérations que les services peuvent réaliser, et cela indépendamment de l'implémentation qui en a été faite. Il ne comporte pas de moyen de décrire de manière plus abstraite les services (tâche plutôt dévolue à DAML-S ou à UDDI), ni de moyen de conversation et de transaction de messages (tel que SOAP ou d'autres implémentations spécifiques), mais est en général utilisé comme passerelle entre ces représentations de haut niveau et de bas niveau.
Dans WSDL, les services sont définis à l'aide de "endpoints". Les "endpoints" sont des ensembles de ports, c'est-à-dire d'adresses sur le réseau associées à certains protocoles et formats de données. Cela va permettre de fournir un cadre abstrait et indépendant des implémentations pour les communications avec les services.
Il y a quatre types d'opérations de base définies dans WSDL : «□sens-unique□», «□double-sens□» requête-réponse, «□double-sens□» sollicitation-réponse, et «□sens-unique□» de message de notification. Les messages et les opérations étant définis de manière abstraite, ce qui permet de faire correspondre ces représentations avec des langages plus abstraits (tel que DAML-S) ; de plus, la réutilisation en est simplifiée. Les messages sont typés, mais on ne peut pas définir de contraintes logiques entre les paramètres d'entrées/sorties au sein de WSDL.

3.3.4 DAML-S

DAML-S est un langage de description de services basé sur XML, et utilisant le modèle des logiques de descriptions. Son intérêt est qu'il est un langage de haut niveau pour la description et l'invocation des services web dans lequel la sémantique est incluse, contrairement par exemple à UDDI. DAML-S est composé de trois parties principales :

- *Service Profile,* qui permet la description, la promotion et la découverte des services, en décrivant non seulement les services fournis, mais également des préconditions à la fourniture de ce service, comme «□avoir une carte bleue

valide» ou «être membre d'un des pays de l'Union Européenne». Les recherches sur les services peuvent se faire en prenant n'importe quel élément de Service Profile comme critère.

- *Service Model,* qui présente le fonctionnement du service en décrivant dans le détail et de manière relativement abstraite les opérations à effectuer pour y accéder. Certains éléments du Service Model peuvent être utilisés à la manière du Service Profile afin de fournir des informations supplémentaires à un utilisateur pour qui les opérations à effectuer seraient également un critère de choix. C'est le Service Model qui va permettre une composition des services si besoin est. Il permet également d'effectuer un contrôle poussé du déroulement du service.
- *Service Grounding* va présenter clairement et dans le détail la manière d'accéder à un service. Tout type abstrait déclaré dans le Service Model s'y verra attribué une manière non ambiguë d'échanger l'information. C'est dans cette partie que le protocole et les formats des messages entre autres sont spécifiés.

Pour l'instant, DAML-S est un langage qui est encore en cours de spécification, mais dont les grandes lignes sont déjà tracées. Un moyen de l'interfacer avec WSDL a été proposé afin de pallier son absence de gestion d'échange de messages, ce qui permettra par exemple d'utiliser SOAP pour échanger des messages XML. DAML-S pourra alors être réservé à une description abstraite et sémantique des services, permettant également d'exprimer des contraintes sur les paramètres et d'utiliser des constructeurs (comme «si…alors…sinon…»).

3.3.5 XL

XL est une plateforme destinée aux services web, axée sur XML, utilisant un langage propre de haut niveau (XL), et prenant en compte les technologies du W3C (WSDL, SOAP) afin de permettre une interopérabilité des applications XL avec d'autres applications écrites dans un langage autre que XL. Tout service web est considéré comme une entité recevant des messages XML et transmettant en retour des messages XML, avec (achat d'un livre) ou sans (consultation de la météo) modification du monde. Les types de données utilisés sont ceux de XQuery, développé lui aussi par le W3C, est dont est inspiré la syntaxe de XL.

La principale motivation de XL est de créer une plateforme qui permette aux programmeurs d'implémenter rapidement des services web en permettant une réutilisabilité maximale. Le langage de requête est un langage déclaratif (à la manière de SQL) et peut donc être optimisé de manière automatique. De plus, comme ce langage est de haut niveau, il permet une composition facilitée des services. XL intègre également une politique de sécurité basée sur J2EE (Java 2 Enterprise Edition), et met l'accent sur le traitement des instructions en mode pipeline, afin d'être plus réactif face à des sources XML importantes ou continues.

Cependant, même si XL permet de manipuler relativement facilement des services web, il ne permet pas de les décrire autrement que par des entrées/sorties XML, et la sémantique est absente, contrairement à DAML-S par exemple. Bien que ce système soit encore en phase de spécification, un prototype de démonstration a été implémenté.

3.3.6 ebXML

Standard fortement industriel et spécifiquement orienté commerce électronique, ebXML (Electronic Business XML initiative) est développé depuis 1999 et se veut un système de standardisation des échanges pour le commerce inter-entreprise (alors qu'UDDI serait plus proche d'un répertoire).

Il s'agit principalement d'un ensemble de protocoles, devenus des standards en 2001, s'appuyant sur des outils tels que CPP (un équivalent de WSDL amélioré), R&R (Registery & Repository, fonctionnant comme UDDI) ou encore SOAP (pour le transfert des messages).
ebXML est vu par beaucoup comme un successeur des EDI (Electronic Data Interchange) qui permettent depuis plusieurs années aux entreprises d'échanger des données de façon sécurisée et fiable, mais qui sont souvent considérées comme des solutions chères et difficiles à mettre en place. ebXML se veut simple à implémenter et généraliste, contrairement à d'autres organismes dédiés à certains groupes d'industries comme RosettaNet (électronique et semi-conducteurs) ou Open Travel Alliance (Compagnies aériennes, location de voitures, hôteliers, agences de voyages…).

3.3.7 RosettaNet

RosettaNet possède de nombreuses similarités avec ebXML. Cependant, bien qu'ebXML corresponde à une conception plutôt horizontale des services web, RosettaNet en a une vision verticale, c'est-à-dire que les informations échangées le seront (entre plusieurs entreprises ou au sein d'une même entreprise) en général concernant un produit ou un ensemble de produits bien particuliers. RosettaNet regroupe actuellement plus de 400 acteurs majeurs de l'industrie de l'électronique et des semi-conducteurs, ainsi que des SSII et des entreprises de Technologies de l'Information.
RosettaNet se base sur des "Partner Interface Processes" (PIPs) qui sont des messages XML échangés par les systèmes et qui définissent la marche à suivre pour commercer entre les partenaires. Chaque PIP inclut un document contenant le vocabulaire employé afin d'éviter tout malentendu (le "business document"), ainsi qu'un autre définissant la chorégraphie de l'échange de messages, c'est-à-dire la façon dont les messages nécessaires au bon déroulement du commerce seront envoyés et reçus (le "business process"). Les PIPs sont utilisés dans des secteurs aussi variés que l'administration, la recherche de partenaires et de produits, la gestion de l'inventaire, la gestion marketing, ou encore le service après-vente.
Un des grands avantages de RosettaNet sur ebXML est que plusieurs solutions ont été implémentées avec succès en entreprise et fonctionnent actuellement en phase d'exploitation. Les similarités entre les deux systèmes sont cependant assez fortes, d'autant plus que RosettaNet a adopté le système de transfert de messages (basé sur SOAP) d'ebXML. Tout indique un rapprochement futur entre ces deux standards.

DAML-S est la seule solution proposant une réelle sémantique des données, et pas seulement des champs prédestinés par la structure des standards ou par des «□euilles de styles□ utilisées pour décrire les services ; de plus, son utilisation des logiques de descriptions pour modéliser les services permet une grande puissance d'expression, que ne possèdent pas les autres systèmes.

4 Recherches futures pour le web sémantique

Le travail sur les langages du web sémantique n'en est qu'à son début. Ces langages devront passer le crible des applications pour déterminer s'ils doivent être amendés ou totalement abandonnés. En attendant, ils laissent derrière eux un certain nombre de questions qui méritent de plus amples recherches afin de faciliter la compréhension de ce que pourra être le web sémantique.

4.1 Modularisation des langages

Nous avons vu que RDF s'intéresse à des assertions sur les relations entre objets, tandis que OWL s'intéresse à décrire les classes de ces objets. Il s'agit d'un découpage assez naturel, entre connaissances factuelles et les connaissances ontologiques. Cette structuration des connaissances a été apportée à la fois par les logiques de descriptions (A-Box et T-Box) et les graphes conceptuels (graphe et support) par rapport à leur ancêtre commun, les réseaux sémantiques. Au niveau des usages, cette séparation est tout aussi importante : la conception des ontologies relève du domaine d'un spécialiste, tandis que les connaissances factuelles, utilisant une ontologie donnée, sont du ressort d'un utilisateur averti. Il aurait été naturel de cloisonner RDF et OWL suivant ces spécifications, mais le besoin d'augmenter l'expressivité de chacun des langages semble avoir été le plus fort. L'extension de RDF à RDFS mélange dans un même graphe deux niveaux d'abstraction très différents, et ce manque de structuration comme de lisibilité est un des principaux reproches qui avait été fait aux réseaux sémantiques (avec le manque d'une sémantique formelle, défaut auquel RDF remédie). De la même façon, on peut coder en OWL DL des connaissances factuelles qui sont du ressort de RDF. Il y a donc un manque de lisibilité sur les objectifs de ces langages, aggravé par leur multiplicité (RDF, RDFS, OWL LITE, OWL DL, OWL FULL).
Un découpage plus clair entre RDF et OWL aurait permis de développer d'une part des ontologies, et d'autre part des documents RDF dont les ressources seraient des classes ou des propriétés décrites dans un document OWL. Cette utilisation commune des deux langages, pourtant naturelle, n'a pour l'instant pas été étudiée. Même si la sémantique de ce langage RDF+OWL se définit immédiatement par les sémantiques des langages qui le composent, d'importants problèmes théoriques se posent : si des algorithmes sont connus pour raisonner sur des documents RDF (homomorphisme de graphe), et pour raisonner dans certains sous-ensembles de OWL (travail réalisé pour les logiques de descriptions), la juxtaposition RDF+OWL n'a pas été étudiée (des problèmes équivalents peuvent se retrouver en BD pour l'inclusion de requêtes, mais reste un travail en cours). Même le découpage de OWL (LITE, DL, FULL) pourrait être remis en cause par la complexité des sous-langages de RDF+OWL correspondant.
Par ailleurs, les besoins différents des utilisateurs peuvent nécessiter un découpage différent de celui retenu par le W3C. Ainsi, on pourrait préférer à OWL DL un langage dont la disjonction est exclue et la sémantique de la négation intuitionniste et où les classes peuvent être considérée comme des instances. Un tel langage n'est pas défini actuellement. Pour en bénéficier, il aurait fallu développer une approche plus modulaire des langages du web sémantique pour laquelle des travaux seraient les bienvenus.

4.2 Moteurs d'inférence

Le développement d'outils efficaces pour raisonner dans le web sémantique sera un critère décisif pour l'adoption de tel ou tel langage. Ce sont ces moteurs d'inférence qu'il faudra encapsuler dans des systèmes de requêtes plus évolués afin d'interroger le web et agir sur les réponses obtenues.
Or, pour le plus simple de ces langages (RDF), la subsomption est un problème NP-complet. Des algorithmes efficaces ont pourtant été développés pour calculer les homomorphismes de graphes qui répondent à ce problème (basés sur les améliorations de rétrogression développés pour les réseaux de contraintes). Ces algorithmes permettent, pour donner un ordre de grandeur, de calculer les homomorphismes d'un graphe à 500 sommets dans un graphe à 3000 sommets dans un temps raisonnable (si ces graphes ne sont pas trop denses). Le problème est maintenant tout autre. Même si nous pouvons supposer que l'ordre de grandeur d'un graphe question est de 50 sommets, la base de faits est l'ensemble des documents RDF disponibles sur le web. Il

y a aujourd'hui plus de 3 milliards de pages HTML référencées par Google, et, sans présager du succès de RDF, on peut se demander combien de documents RDF seront disponibles demain. Bien que nous pensons que la réalisation d'algorithmes efficaces soit possible (avec de bons mécanismes d'indexation pour démarrer le raisonnement, car il suffit d'étendre localement des homomorphismes partiels), seule une expérimentation sur une grande masse de données réelles peut permettre de valider cette intuition.
Cependant, dans le langage RDF+OWL que nous jugeons souhaitable, les problèmes deviennent tout autres. Même en n'ajoutant que la négation atomique de type, le problème de subsomption devient Π_2^P-complet. Un traitement local de l'information au cours de l'exécution de l'algorithme n'est alors plus envisageable.

4.3 Transformation de langages

Il y a fort à parier que la connaissance sera disponible sur le web dans des formes (langages) différentes, avec des modèles (ontologies) différents. Qui plus est, certaines applications auront besoin de fusionner de telles sources de connaissance ou de les adapter à leurs besoins. Cette activité est pour l'instant réalisée de manière ad hoc (dans l'écriture de wrappers par exemple). Il sera nécessaire pour tirer pleinement parti de la connaissance disponible dans le web sémantique de la transformer et de l'importer sous des contextes (langages, ontologies) différents. Il existe une grande variété de telles transformations (fusion de catalogues, extraction de bases de données, normalisation de théories) nécessitant diverses propriétés (filtrantes, préservant les conséquences…). Un premier effort de recherche devrait permettre de caractériser ces transformations et leurs relations. Il devrait aussi être possible de définir de manière standard une transformation «□sémantique□» et surtout de l'exécuter. Actuellement, il n'existe aucune infrastructure de transformation pour RDF et l'on utilise toujours XML pour cela.
Enfin, puisque ces opérations sont destinées à être effectuées par des machines (sans discernement), il est essentiel pour la crédibilité du web sémantique que l'on puisse prouver la correction des transformations par rapport à leurs spécifications.

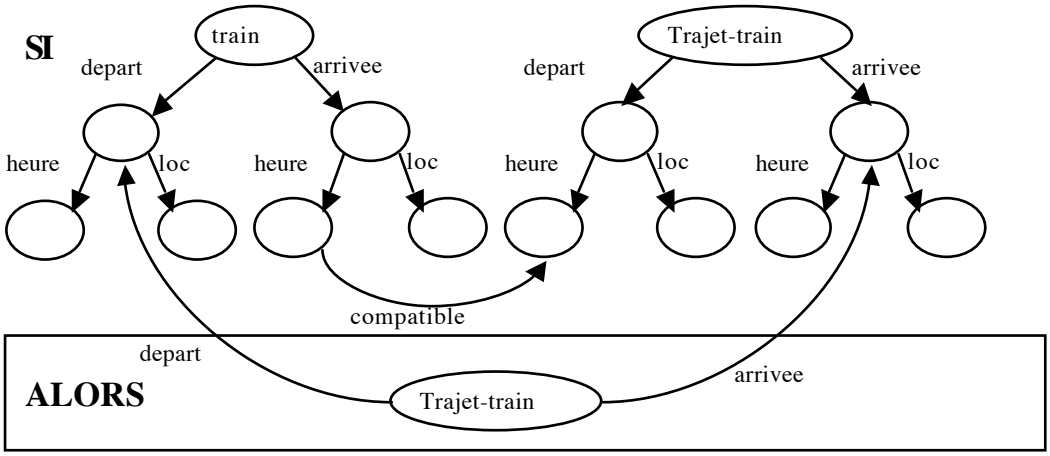
4.4 Inférences robustes

Une propriété typique du web est la quantité d'information que l'on y trouve. Malheureusement, il s'y trouve beaucoup d'information erronée, périmée, redondante ou incomplète. Le surfer humain est en général capable de discerner ces problèmes et de les surmonter sans trop y penser. Mais il n'en ira pas de même des applications du web sémantique. Il est donc nécessaire de développer des modes de raisonnement qui tirent parti du web sémantique, c'est-à-dire qui sont le plus fidèle possible aux spécifications des langages utilisés, sans pour autant être perturbés par ces problèmes. En un mot, il faut des moteurs d'inférence robustes.
Ceux-ci pourraient utiliser des techniques très variées (logiques paraconsistantes ou modèles statistiques, raisonnement non monotone) adaptées au contexte du web sémantique. Le raisonnement anytime ou sous contraintes de ressources pourrait être quant à lui utilisé pour gérer l'immense taille du web.

4.5 Langages de règles

Une autre nécessité, soulignée par de nombreuses personnes travaillant sur le web sémantique, est de développer un langage de règles. Si un organisme *X* déclare sur son document RDF qu'un train va de la ville *A* à la ville *B*, et un organisme *Y* déclare qu'un train va de la ville *B* à la ville *C*, alors il faut en déduire qu'il existe un trajet allant de *A* à *C*. Pourtant, cette information ne peut se trouver ni sur le site de *X*, ni sur

celui de *Y*. Une solution immédiate peut être de déclarer, dans un document OWL, que la propriété trajet est transitive, mais le problème devient insoluble dès lors que l'on veut prendre en compte l'existence ou la durée de la correspondance. Il est nécessaire d'utiliser une règle «□SI … ALORS□.» que l'on pourrait représenter de la façon suivante□:



Ce type de règle a été étudié comme une extension des graphes conceptuels simples, et les résultats obtenus sont immédiatement transférables à une extension de RDF. Ces règles sont dotées d'une sémantique, qui correspond à des formules logiques de la forme□:

$$\forall X\,(P(X) \rightarrow (\exists Y\, Q(X,Y)))$$

où *P(X)* est une conjonction de formules atomiques dont les variables sont celles apparaissant dans *X*, et *Q(X, Y)* est une conjonction de formules atomiques dont les variables sont celles apparaissant dans *X* et *Y*. Notons que ces formules correspondent aux TDGs (Tuple Generating Dependencies) en bases de données. L'utilisation de telles règles génère un langage très expressif (puisqu'il s'agit d'un modèle de calcul), et malheureusement indécidable. Des sous-ensembles décidables (et même NP-complets) intéressants ont été exhibés.
Il est à noter que l'expressivité d'un tel langage de règles en ferait un bon candidat pour un métalangage permettant, par exemple, de doter un langage de définition d'ontologies de nouveaux constructeurs, en définissant de manière opérationnelle leur sémantique.

Ces travaux, comme d'ailleurs tous les travaux sur les langages, ne peuvent se faire de manière isolée. Il est donc normal qu'ils soient poursuivis en liaison avec les groupes de travail internationaux contribuant à faire avancer l'état de l'art. Par contre un effort conséquent pourrait être produit par une communauté restreinte en ce qui concerne les moteurs d'inférence et de transformation. Mais cela demande un investissement important à moyen terme.

5 URLs

UDDI Universal Description Discovery and Integration of Web services
http://www.uddi.org
WSDL Web Service Description Language
http://www.w3.org/TR/wsdl
DAML-S DARPA Agent Markup Language - Services

http://www.daml.org/services
RosettaNet,
http://www.rosettanet.org
SOAP
http://www.w3.org/TR/SOAP
XL XML Language
http://xl.in.tum.de
ebXML e-Business XML
http://www.ebxml.org
WPDL Workflow Process Definition Language
http://www.wfmc.org/standards/docs/TC-1016-P_v11_IF1_Process_definition_Interchange.pdf
Topic Maps
http://www.y12.doe.gov/sgml/sc34/document/0129.pdf
OWL Web Ontology Language
http://www.w3.org/2001/sw/WebOnt
RDF Resource Description Framework
http://www.w3.org/2001/sw/RDFCore
DAML+OIL DARPA Agent Markup Language + Ontology Inference Language
http://www.daml.org/2001/03/daml+oil-index
KIF Knowledge Interchange Format
http://logic.stanford.edu/kif/
XML eXtensible Markup Languages
http://www.w3.org/TR/REC-xml
XTM XML Topic Maps
http://www.topicmaps.org/xtm/1.0/
XPDL XML Process Definition Language