

Étude bibliographique

Construction et utilisation des logiques dans les systèmes d'information

Étienne André*

1^{er} février 2007

Résumé

Le nombre et la diversité des documents électroniques ne cessent d'augmenter. Cela pose le problème de l'organisation et de la recherche dans un ensemble de documents. Des formalismes logiques peuvent être employés mais, en raison de la diversité des formats de documents, il est souhaitable de pouvoir adapter le formalisme logique en fonction de l'application. Cette étude bibliographique présente deux types de systèmes d'information : le Web sémantique, qui s'appuie sur les logiques de descriptions, et les systèmes d'information logiques, qui s'appuient sur les foncteurs logiques. Ces deux types de systèmes d'information sont présentés et comparés vis-à-vis de critères d'expressivité, de connaissances du domaine et de généricité.

* M2R informatique filière méthodes formelles à l'Ifsic (eandre@irisa.fr)

Table des matières

1	Introduction	3
2	Web sémantique	3
2.1	Les logiques de descriptions	4
2.1.1	Syntaxe	4
2.1.2	Exemple	5
2.1.3	Subsorption et instance	6
2.1.4	Extensions	6
2.1.5	Complexité	7
2.2	Expressivité	7
2.2.1	XPath	8
2.2.2	Raisonnement grâce aux logiques de descriptions	8
2.3	Connaissances du domaine	9
2.4	Généricité	9
3	Systèmes d'information logiques	9
3.1	Présentation	9
3.1.1	Analyse de concepts logique	10
3.1.2	Camelis	10
3.2	Les foncteurs logiques	11
3.2.1	Présentation	11
3.2.2	Exemples	12
3.2.3	Implémentation	13
3.3	Expressivité	13
3.4	Connaissances du domaine	14
3.5	Généricité	14
3.5.1	Composition de logiques modales	14
3.5.2	Avantages des foncteurs logiques	15
4	Conclusion	15

1 Introduction

Le nombre et la diversité des documents électroniques ne cessent d'augmenter. Cela pose le problème de l'organisation et de la recherche dans un ensemble de documents. Une solution courante est de proposer un langage de requêtes, par exemple des listes de mots dans le cas des moteurs de recherche, ou encore SQL pour les bases de données. Des formalismes logiques peuvent être employés pour la représentation des requêtes et des données d'une part, et pour la définition des relations complexes entre requêtes et réponses par déduction d'autre part [Rij87]. En raison de la diversité des formats de documents, il est souhaitable de pouvoir adapter le formalisme logique en fonction de l'application, ce qui implique un besoin de généralité.

Dans le monde de la recherche, une même personne tend souvent à avoir plusieurs rôles : on peut être simultanément théoricien, développeur d'applications et même utilisateur final. Nous ferons désormais référence à cette notion de rôles comme des « rôles-utilisateurs »¹. Cependant, cette vision n'est pas systématiquement envisageable. Les utilisateurs finaux ne développent en général pas leurs applications, et les développeurs d'applications n'ont pas toujours des compétences théoriques pointues. Le développeur d'une application susceptible d'analyser une grande quantité de documents doit donc pouvoir choisir une logique appropriée pour l'accès à ces documents, sans avoir à développer les outils ayant trait à cette logique, lesquels requièrent une expertise logique.

Cette étude bibliographique présente deux types de systèmes d'information s'appuyant sur la logique et permettant d'indexer et de rechercher des documents. Tout d'abord, le Web sémantique est présenté en section 2, ainsi que les logiques de descriptions (2.1) sur lesquelles il s'appuie. Puis les systèmes d'information logiques sont présentés en section 3, ainsi que les foncteurs logiques (3.2) sur lesquels ils s'appuient. Ces deux types de systèmes d'information sont présentés et comparés au regard de critères d'expressivité, de connaissance du domaine et de généralité. Enfin, la section 4 propose une synthèse et des pistes de recherche.

2 Web sémantique

Le Web sémantique désigne un cadre commun visant à rendre le contenu des ressources du Web accessible par des programmes, grâce à un système de métadonnées formelles, c'est-à-dire de données permettant de décrire d'autres données. Le Web sémantique est mis en œuvre par le W3C (*World Wide Web Consortium*), chargé de promouvoir la compatibilité des technologies du Web en émettant des recommandations, avec la participation de chercheurs et de partenaires industriels [WS01]. Là où le Web traditionnel utilise les balises HTML [LHRJ99], le W3C recommande notamment dans le Web sémantique d'utiliser le langage de métadonnées XML [CGL99].

XML (*Extensible Markup Language*, « langage de balisage extensible ») est un langage de balisage générique. Son objectif est de faciliter l'échange automatisé de contenus entre différents systèmes d'information, notamment sur Internet [BPSM⁺06]. Des techniques de recherche et d'exploitation de documents

¹Ceci afin d'éviter la confusion entre cette notion de rôle-utilisateurs et les rôles définis dans le cadre des logiques de descriptions.

```

<publications>
  <article ref="DLNN97">
    <auteur>F. M. Donini</auteur>
    <auteur>M. Lenzerini</auteur>
    <auteur>D. Nardi</auteur>
    <auteur>W. Nutt</auteur>
    <titre>The complexity of concept languages</titre>
    <annee>1997</annee>
  </article>
  <these ref="Fer02">
    <auteur>S. Ferré</auteur>
    <titre>Systèmes d'information logiques</titre>
  </these>
</publications>

```

Figure 1 – Exemple de document XML

XML peuvent également connaître d'autres finalités que le Web sémantique : par exemple, la base d'articles scientifiques DBLP², qui contenait en décembre 2006 près de 830 000 documents, est exploitée en XML.

La figure 1 représente un exemple de document XML regroupant 2 références bibliographiques, contenant chacune un ou plusieurs auteur(s), un titre et une année éventuelle. La syntaxe de XML est reconnaissable par la présence de *balises* encadrées par des chevrons < et > — par exemple <publications>.

Il existe plusieurs techniques d'interrogation de documents XML, parmi lesquelles XPath, permettant de poser des requêtes sur des documents XML. XPath sera présenté en section 2.2.1 page 8.

2.1 Les logiques de descriptions

L'interrogation de documents XML peut également se faire grâce aux logiques de descriptions, recommandées par le W3C pour le Web sémantique. Cette section s'inspire de l'introduction aux logiques de descriptions d'Amedeo Napoli [Nap97].

2.1.1 Syntaxe

Les logiques de descriptions sont une classe de logiques permettant de modéliser des connaissances à l'aide de « descriptions » qui peuvent être des *concepts*, des *rôles* ou des *individus*. Un concept permet de représenter un ensemble d'individus, tandis qu'un rôle permet de décrire une relation binaire entre individus. Les individus sont des *instances* des concepts : on distingue alors la manipulation des rôles et des concepts, qui relève du niveau *terminologique* (*TBox*), de la manipulation des individus, qui relève du niveau *factuel* ou *assertionnel* (*ABox*).

L'ensemble des descriptions forme un treillis. Les rôles et concepts primitifs sont introduits par le symbole \leq , tandis que les concepts définis sont introduits par le symbole \doteq .

La syntaxe de la logique de descriptions basique \mathcal{AL} est donnée sur la figure 2. C représente une expression de concept, A un nom de concept primitif

²<http://dblp.uni-trier.de/>

ARTICLE (Fer02)
 CHERCHEUR (Ferré)
 titre (Fer02, *Systèmes d'information logiques*)
 auteur (Fer02, Ferré)
 annee (Fer02, 2002)

Figure 4 – Exemples d'assertions

Extension	Notation
Négation de concepts non nécessairement primitifs	$\mathcal{ALC} = \mathcal{AL} \cup \{-C\}$
Disjonction de concepts	$\mathcal{ALU} = \mathcal{AL} \cup \{C \sqcup C\}$
Quantification existentielle typée	$\mathcal{ALE} = \mathcal{AL} \cup \{\exists r.C\}$
Cardinalité sur les rôles	$\mathcal{ALN} = \mathcal{AL} \cup \{\geq n r, \leq n r\}$
Conjonction de rôles	$\mathcal{ALR} = \mathcal{AL} \cup \{r \sqcap r\}$

Figure 5 – Exemples d'extensions de la logique \mathcal{AL}

Le concept **ARTICLE-COMMUN** est un concept défini, représentant l'ensemble des articles dont tous (\forall) les auteurs sont des chercheurs et (\sqcap) ayant au moins (\geq) deux auteurs. Le \sqcap permet la conjonction de concepts, le \forall définit le codomaine d'un rôle, tandis que le \geq précise la cardinalité d'un rôle en indiquant le nombre minimal de valeurs élémentaires du rôle. Le concept **PUBLI-NON-ANONYME** est également un concept défini, correspondant aux publications ayant au moins un auteur.

Les exemples d'assertions de la figure 4 expriment la définition d'un article Fer02 ainsi que celle d'un chercheur. Cet article est associé à un titre par le rôle **titre**, à ce chercheur (rôle **auteur**) et à une année (rôle **annee**).

2.1.3 Subsumption et instance

Les relations clés des logiques de descriptions sont la relation de *subsumption* et la relation « être instance de ». Tout d'abord, la relation de subsumption permet d'organiser concepts et rôles par généralité : intuitivement, un concept D est subsumé par un concept C (noté $D \sqsubseteq C$) si C est plus général que D , c'est-à-dire si l'ensemble d'individus représenté par C contient l'ensemble d'individus représenté par D dans toutes les interprétations. Par exemple, le concept **PUBLICATION** subsume le concept **ARTICLE** et, de manière moins triviale, le concept **PUBLI-NON-ANONYME** subsume le concept **ARTICLE-COMMUN**. Enfin, la relation « être instance de » permet de connaître les individus instances d'un concept. Par exemple, Fer02 est une instance des concepts **ARTICLE** et **PUBLI-NON-ANONYME**.

2.1.4 Extensions

La logique \mathcal{AL} définie ci-dessus représente la base minimale des logiques de descriptions. Elle peut être enrichie par plusieurs constructeurs en fonction des besoins. La figure 5 présente les extensions amenant la négation de concepts non nécessairement primitifs, la disjonction de concepts, la quantification existentielle typée, la notion de conjonction et de cardinalité sur les rôles. Par

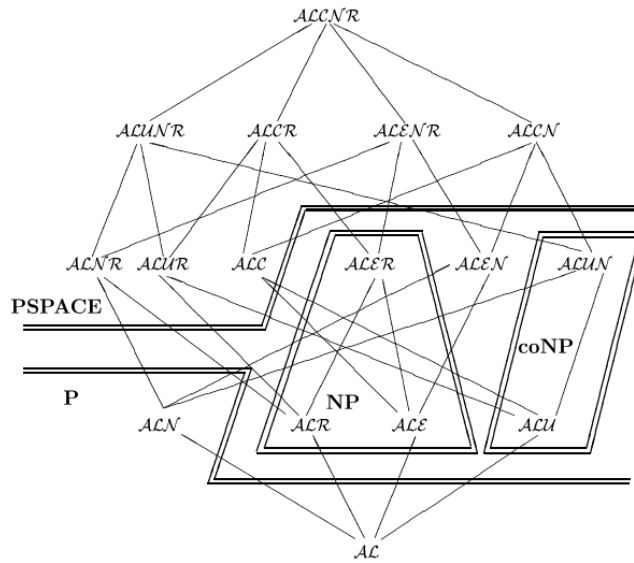


Figure 6 – Treillis de logiques de descriptions

combinaison de ces constructeurs, on peut ainsi définir de nouvelles logiques de descriptions. On notera que certaines extensions en impliquent d'autres : par exemple, l'union (\mathcal{U}) et la quantification existentielle typée (\mathcal{E}) sont présentes dans toute logique implémentant la négation (\mathcal{C}), et inversement. Par conséquent, $\mathcal{ACUEN} = \mathcal{ACCN}$. Ces logiques de descriptions forment un treillis présenté sur la figure 6 [DLNN97].

2.1.5 Complexité

Ces logiques sont toutes décidables, mais de complexité différente. L'intérêt de raisonner avec le minimum d'extensions nécessaire est de limiter la complexité, ce qui a son importance lorsqu'on raisonne sur de gros volumes d'information. La complexité est fonction des extensions choisies ; la figure 6 présente la complexité des relations de non-satisfiabilité et de subsomption dans les logiques de descriptions de la famille \mathcal{AL} . Pour fixer les idées, cette complexité est polynomiale pour \mathcal{AL} et \mathcal{ALN} , mais devient NP (non-déterministe polynomiale) ou co-NP lors de l'ajout de certaines extensions, et devient rapidement PSPACE, c'est-à-dire décidable par un algorithme déterministe en espace polynomial par rapport à la taille de son instance, lorsque plusieurs extensions sont ajoutées.

Un autre exemple de logique de descriptions est la logique \mathcal{SHOIQ} , qui a entre autres comme spécificité la notion de rôle inverse. Étant donné un ensemble de noms de rôles N_R , un rôle est soit un rôle $R \in N_R$, soit un rôle inverse R^- où $R \in N_R$ [HS05]. Intuitivement, **enfant** est le rôle inverse de **parent**.

2.2 Expressivité

L'expressivité définie dans cette section se réfère à la facilité qu'a un système à représenter, d'une part, des données et, d'autre part, des requêtes. Cependant,

un compromis doit généralement être fait entre expressivité et efficacité, puisque l'une est souvent incompatible avec l'autre. Les applications n'étant pas *a priori* figées, une certaine flexibilité est souhaitée afin de pouvoir satisfaire l'une ou l'autre de ces contraintes selon les besoins.

La structure des documents du Web sémantique est basée sur XML, un méta-langage permettant une analyse syntaxique automatique à partir d'un schéma spécifié par une DTD (définition du type de document) ou XML Schema. L'une des limites de l'expressivité de XML réside dans le fait que, par sa structure-même, il impose la notion d'arbre. L'exemple présenté figure 1 page 4 montre la structure d'arbre d'un document.

2.2.1 XPath

Les requêtes sur un document XML ou une liste de documents XML modélisée par un unique document peuvent être exprimées grâce au langage XPath. Celui-ci permet de retrouver des parties d'un document XML [XP99].

Pour fixer les idées, sans détailler la syntaxe de XPath, / permet de représenter un chemin relatif, tandis que // représente zéro, un ou plusieurs pas d'un chemin. @ permet d'accéder aux attributs d'un élément (par exemple ref dans l'exemple 1). Enfin, l'astérisque * permet de considérer tous les fils d'un nœud. Quelques exemples d'accès à des parties du document de la figure 1 sont donnés ci-dessous [BFGHK05] :

- /publications permet d'accéder à l'ensemble des publications
- /publications/*/* permet d'accéder aux petits-enfants de la racine
- //article permet d'accéder à l'ensemble des articles
- //these[titre="Systèmes d'information logiques"] permet d'accéder à l'ensemble des thèses ayant le titre considéré (c'est-à-dire dans notre cas la thèse dont l'attribut ref est Fer02)
- //article/publication ne rendra rien du tout car n'est pas conforme à la structure du document défini

2.2.2 Raisonnement grâce aux logiques de descriptions

Une autre manière de représenter et de raisonner sur des documents XML est d'utiliser les logiques de descriptions. Les logiques de descriptions permettent en effet, d'une part, de représenter les DTD et, d'autre part, d'exprimer des requêtes pour déterminer l'ensemble des documents vérifiant un ensemble de propriétés à partir d'une liste de documents XML [CGL99]. Dans ce cas, la relation d'inclusion entre éléments XML ainsi que les attributs de XML sont codés par des rôles simples dans les logiques de descriptions, et les balises du langage XML sont représentées par des concepts primitifs. L'expressivité des requêtes sur des documents du Web sémantique est alors celle des logiques de descriptions.

Les logiques de descriptions peuvent être vues comme un intermédiaire entre la logique propositionnelle et la logique du premier ordre. Les logiques de descriptions sont similaires à la logique propositionnelle à qui l'on aurait ajouté la notion de rôles, mais sont également un fragment de la logique du premier ordre, où les rôles constituent des prédicats binaires et les concepts des prédicats unaires, et où seuls deux noms de variables sont utilisés. Une traduction de la logique de descriptions *SHOIQ* vers la logique du premier ordre a été

proposée [KM06]. Différentes extensions de la logique \mathcal{AL} , dont certaines sont présentées figure 5 page 6, permettent de prendre en compte les notions de quantification existentielle, de négation, de disjonction ou de rôle inverse.

2.3 Connaissances du domaine

OWL (*Web Ontology Language*) est un langage d'ontologies recommandé par le W3C dans le cadre du Web sémantique pour faciliter l'analyse de l'information par des programmes et non uniquement par des humains. La logique de descriptions *SHOIQ* est notamment à la base de OWL. OWL est devenu *de facto* le standard le plus utilisé pour les ontologies. OWL apporte un vocabulaire et des contraintes, permettant de définir des propriétés telles que la cardinalité (par exemple « au moins un »), l'égalité, les relations entre concepts (disjonction), etc. OWL permet donc de définir des terminologies pour décrire des domaines d'application et peut être vu comme le pendant pour le Web sémantique de la *TBox* des logiques de descriptions.

2.4 Généricité

Le Web sémantique est intrinsèquement générique du fait qu'il se base sur le format XML constituant un cadre personnalisable selon les applications. En revanche, se pose la question de la réutilisabilité des composants du Web sémantique pour un développeur d'applications.

Une technique de composition de services Web a été proposée, basée sur la notion d'ontologie [MBE03]. Cette technique détermine une description syntaxique et sémantique des services ainsi que des propriétés telles que le type de leurs paramètres. Puis la *composabilité* des services Web est vérifiée à partir de ces propriétés. Cette technique propose ensuite une composition automatisée : plusieurs possibilités de composition sont étudiées et une composition est sélectionnée grâce à des critères valués de complétude et pertinence notamment.

On notera que les propriétés (type des paramètres) sont définies sur le plan syntaxique uniquement, alors que l'on pourrait souhaiter considérer des propriétés plus complexes, comme la complétude ou la consistance des opérations. En outre, seule une spécification du résultat est ici donnée, charge au système de trouver les éléments composables, alors qu'il pourrait être souhaitable de donner explicitement la combinaison souhaitée.

3 Systèmes d'information logiques

Les systèmes d'information logiques (*logical information systems*) utilisent les logiques de manière uniforme pour décrire des collections d'objets, effectuer des requêtes, naviguer parmi ces objets, les analyser et les maintenir. Les systèmes d'information logiques sont développés par l'équipe LIS à l'Irisa [Fer02, PR03, FR04, BFRQ06]. Le stage découlant de cette étude bibliographique sera effectué au sein de cette équipe.

3.1 Présentation

À la différence de nombreux systèmes d'indexation et de recherche de données qui font un choix exclusif entre le schéma requêtes – réponses (par exemple

SQL) ou le parcours d'une hiérarchie (par exemple les systèmes de fichiers), les systèmes d'informations logiques lient interrogation et navigation. Ainsi, l'utilisateur peut poser des requêtes puis naviguer parmi les résultats, le système lui proposant des incréments de navigation. Inversement, l'utilisateur peut également naviguer dans un système d'information logique, et le système lui propose des incréments de requête. On peut ainsi utiliser alternativement ou simultanément les deux techniques.

3.1.1 Analyse de concepts logique

Les systèmes d'information logiques sont basés sur la théorie de l'analyse de concepts logique [FR04], elle-même basée sur l'analyse de concepts formelle [GW99]. Un concept formel est l'association entre un ensemble d'objets et une formule. Les objets relatifs à un concept sont son *extension*, tandis que la formule que le concept représente est son *intension*. Ainsi, concepts, requête et endroit de navigation sont fusionnés. L'ensemble des concepts forme un treillis, ce qui définit une structure de navigation ajustée en permanence aux données. Héritant des propriétés de l'analyse de concepts formelle, les systèmes d'information logiques ont tout d'abord représenté des attributs simples. Par la suite, dans un souci d'expressivité, un mouvement vers la notion de logique s'est effectué. Les systèmes d'information logiques ont permis l'emploi de la logique propositionnelle (avec les connecteurs logiques \wedge , \vee , \neg), puis d'attributs valués, comme les chaînes de caractères ou les intervalles.

Les systèmes d'information logiques s'appuient sur une logique arbitraire \mathcal{L} et une relation \sqsubseteq de subsomption. Les descriptions d'objets sont représentées par une formule logique d , et les requêtes par une formule q ; ainsi, pour déterminer si un objet répond à une requête, il suffit de vérifier que la formule de description de l'objet est subsumée par la formule de la requête $d \sqsubseteq q$.

3.1.2 Camelis

La figure 7 montre une capture d'écran de Camelis, le logiciel de gestion de documents basé sur les systèmes d'information logiques³, à même de gérer l'indexation et la recherche de documents MP3, de photos ou de tout autre type de fichier, grâce à la généralité de la logique. La dualité entre interrogation et navigation est visible d'une part grâce à la requête `Animal and Oceanie` affichée dans la partie supérieure et, d'autre part, grâce aux liens de navigation sur la partie gauche. L'endroit courant de la recherche concerne l'ensemble des photos d'animaux prises en Océanie; le système propose alors des incréments de requête par date (21 photos ont été prises en février 2004, 13 en mars 2004), par espèce (mammifères, oiseaux, etc.) ou par d'autres critères. Si l'on clique sur l'un des liens de navigation, la requête se précise et le système propose à nouveau des liens de navigation plus pertinents.

Un exemple de requête dans Camelis est présenté ci-dessous :

```
(titre contains "Logical Information Systems"
OR titre contains "Systèmes d'information logiques")
AND NOT auteur is "Ferré"
```

³<http://www.irisa.fr/lande/ferre/camelis/>

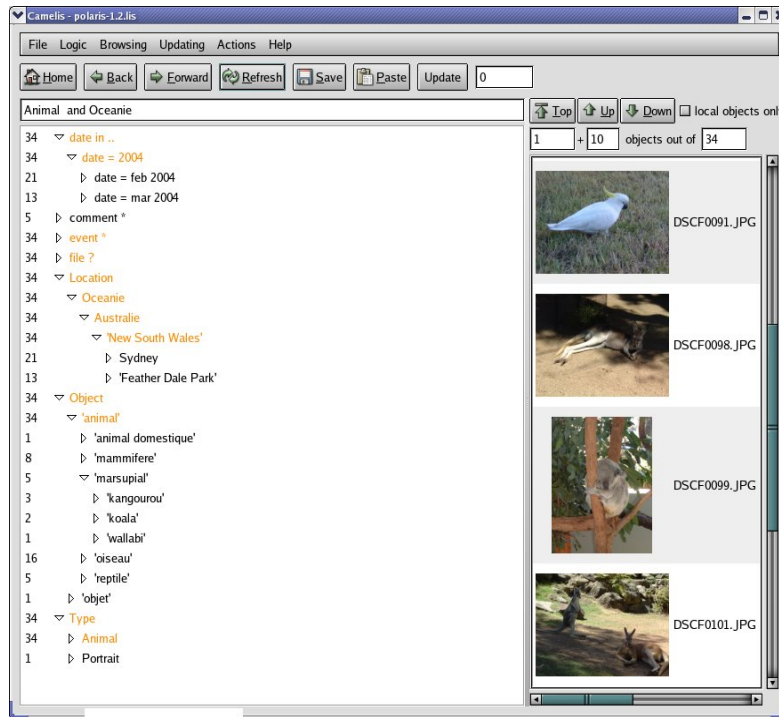


Figure 7 – Capture d'écran de Camelis

Cette requête va retourner l'ensemble des publications dont le titre contient « systèmes d'information logiques » (en anglais ou en français) et dont l'auteur n'est pas « Ferré ».

3.2 Les foncteurs logiques

Le fait que différents rôles-utilisateurs (comme défini en introduction) puissent participer à la création puis à l'utilisation d'un système d'information logique nécessite une genericité de l'application vis-à-vis de sa logique, ainsi que la possibilité pour le développeur d'applications de personnaliser la logique utilisée — ce que permettent les foncteurs logiques [FR02, FR06].

3.2.1 Présentation

Les foncteurs logiques ont été développés à l'origine pour répondre aux besoins de genericité de la logique dans les systèmes d'information logiques. Les foncteurs logiques permettent la combinaison d'éléments primitifs de logiques, afin de créer de nouvelles logiques, de manière analogue à un jeu de LegoTM. L'idée des foncteurs logiques est de considérer qu'une logique interprète ses formules comme une fonction de ses atomes. En réalisant une abstraction des formules atomiques du langage d'une logique, nous obtenons donc un foncteur logique.

Une logique est constituée d'une syntaxe abstraite, d'une sémantique, d'une implémentation et d'un type. La sémantique est similaire à celle des logiques

de descriptions : en effet, la sémantique d'une formule est l'ensemble d'interprétation de cette formule (l'ensemble des objets vérifiant cette formule). L'implémentation est un ensemble d'opérations sur la syntaxe abstraite. Parmi ces opérations, la principale est le test de subsomption \sqsubseteq , permettant d'ordonner les formules selon une plus ou moins grande généralité. Enfin, le type d'une logique représente le certificat d'une logique ; il s'agit de l'ensemble des méta-propriétés vérifiées par cette logique, par exemple la consistance ou la complétude de la relation de subsomption vis-à-vis de la sémantique. La classe de toutes les logiques est notée \mathbb{L} .

Un foncteur logique prend une ou plusieurs logique(s) en paramètre(s), et génère une nouvelle logique. Les foncteurs logiques ont donc également une syntaxe, une sémantique, une implémentation et un type, mais tous sont abstraits sur une ou plusieurs logiques jouant le rôle de paramètres formels. Ainsi, la logique $Prop(X)$ est la logique propositionnelle, dont les atomes sont abstraits par le paramètre formel X et remplaçables par des formules plus complexes. On peut alors construire $Prop(Atom)$, la logique propositionnelle classique, ou encore $Prop(Interval(Int))$, en remplaçant les atomes par des intervalles d'entiers.

Formellement, la classe des foncteurs logiques \mathbb{F} inclut les fonctions de \mathbb{L}^n dans \mathbb{L} . Le cas particulier des foncteurs logiques d'arité $n = 0$ correspond à la classe des logiques \mathbb{L} . La notation F/n signifie que le foncteur logique F est d'arité n , et est donc fonction de n logiques.

La syntaxe d'un foncteur logique est simplement fonction des syntaxes des logiques passées en paramètres. Il en va de même pour sa sémantique, son implémentation et son type.

3.2.2 Exemples

Quelques exemples de foncteurs logiques sont détaillés ci-dessous. Ceux-ci sont classés vis-à-vis de leur rôle dans la composition des logiques.

Initiateurs Foncteurs logiques constants (arité $n = 0$) représentant divers domaines concrets :

- $Atom/0$: les atomes utilisés classiquement,
- $Int/0$: les entiers,
- $String/0$: les chaînes de caractères et des opérations telles que « contient » ou « commence par ».

Constructeurs Foncteurs logiques définissant la structure des formules et des interprétations :

- $Sum/2$: l'union disjointe de deux logiques,
- $Prod/2$: le produit de deux logiques,
- $Multiset/1$: multi-ensemble sur les formules de la logique.

Abstracteurs Foncteurs logiques étendant la syntaxe abstraite sans changer le domaine d'interprétation :

- $Interval/1$: intervalles sur une logique de valeurs ordonnées,
- $Prop/1$: fermeture de la syntaxe par les trois connecteurs booléens (\wedge, \vee, \neg).

Adaptateurs Foncteurs logiques assurant certaines méta-propriétés :
– *Set/1* : application de la notion d'ensemble au domaine d'interprétation.

3.2.3 Implémentation

Les foncteurs logiques sont implémentés sous forme de foncteurs Caml. Si la définition d'un foncteur logique requiert des connaissances en Caml et en logique, la composition de foncteurs logiques et la création d'une nouvelle logique ne requièrent aucune connaissance, ni en Caml, ni en logique. L'exemple ci-dessous montre une composition de foncteurs logiques :

$$Prop(Set(Prod(Atom, Sum(Interval(Int), String))))$$

On raisonne ici sur une logique propositionnelle (foncteur logique *Prop*) d'ensembles (*Set*) de couples (*Prod*) constitués, d'une part, d'éléments atomiques (*Atom*) et, d'autre part, d'intervalles d'entiers (*Interval(Int)*) ou (*Sum*) de chaînes de caractères (*String*). L'une des applications peut ainsi être la représentation d'entrées BibTeX, qui forment effectivement un ensemble de couples constitués d'un atome, correspondant à un nom d'attribut, et d'une chaîne de caractères ou d'un intervalle d'entiers, correspondant à une valeur.

Les foncteurs logiques fournissent en outre une aide au diagnostic, permettant de visualiser les propriétés manquantes à la logique construite, ainsi qu'un certificat assurant que le test de subsomption \sqsubseteq est correct et complet. Enfin, la technique des foncteurs logiques fournit également un analyseur syntaxique (*parser*) et un afficheur (*printer*) afin de traiter la syntaxe concrète des formules.

3.3 Expressivité

L'expressivité dans les systèmes d'information logiques dépend de la logique choisie, et donc des foncteurs logiques utilisés par l'application.

Les foncteurs logiques initiateurs permettent de représenter plusieurs domaines concrets (entiers *Int*, chaînes de caractères *String*, etc.) et les opérations correspondantes (l'égalité entre entiers, la relation d'inclusion d'une chaîne dans une autre, etc.).

En outre, les foncteurs logiques constructeurs permettent la représentation de termes algébriques, tels que les tuples (foncteur logique *Prod*) ou les types somme (foncteur logique *Sum*), ainsi que les opérations de la logique propositionnelles (foncteur logique *Prop*).

Il existe par ailleurs un foncteur logique de point fixe permettant la récursion. Grâce à cette possibilité d'effectuer des définitions récursives de foncteurs logiques, la recomposition de la logique de descriptions *ALC* est possible.

$$ALC = Prop(Set(Sum(Atom, Prod(Atom, ALC))))$$

La logique *ALC* est en effet la logique propositionnelle (*Prop(Set)*) où les atomes sont soit (*Sum*) des concepts atomiques (*Atom*), soit des couples (*Prod*) formés d'un atome *r* (*Atom*) et d'une expression conceptuelle *C* (foncteur logique récursif *ALC*), ce qui correspond à une expression $\exists r.C$. La quantification universelle est représentable par la combinaison de la négation et de la quantification existentielle : $\forall r.C = \neg \exists r. \neg C$ [FR06].

On note ici que la recombinaison de la logique ALC est l'une des applications les plus avancées des foncteurs logiques, alors que cette logique est à l'inverse l'une des plus basiques des logiques de descriptions. Il reste par conséquent une marge de manœuvre importante du côté de la recombinaison de logiques de descriptions par des foncteurs logiques. En effet, représenter une logique comme $SHOIQ$ semble nettement moins aisé : d'une part, la gestion des rôles inverses ou des rôles quantifiés n'a pas encore été effectuée dans les foncteurs logiques et, d'autre part, la notion de terminologie (connaissances du domaine) n'a pas été considérée à l'heure actuelle.

Enfin, l'efficacité d'un système d'information logique dépend, comme pour son expressivité, de la logique choisie. Si raisonner sur la logique atomique $Atom$ est très efficace, du fait que la relation de déduction est une simple égalité, la logique propositionnelle $Prop(Set(Atom))$ est, elle, nettement plus coûteuse. Pour néanmoins fixer les idées, une fois la logique et la taille des descriptions d'objets fixées, l'efficacité d'un système d'information logique est, en fonction de l'opération effectuée, soit constante, soit linéaire avec le nombre d'objets.

3.4 Connaissances du domaine

La connaissance du domaine est câblée dans les foncteurs logiques. Mais se pose alors le problème de la distinction des rôles-utilisateurs évoquée en introduction. En effet, le logicien définit alors la logique *et* la terminologie, alors qu'une distinction souhaitable serait la suivante : le logicien développe les foncteurs logiques de base, un développeur les combine, un expert du domaine définit une ontologie, et un utilisateur se sert de l'application ainsi définie.

3.5 Généricité

3.5.1 Composition de logiques modales

Certains travaux ont été effectués sur les compositions de logiques, en particulier sur les logiques modales, qui ont de fortes similitudes avec les logiques de descriptions [Sch91]. Un ensemble de traits de ces logiques a été défini et, en sélectionnant un certain nombre de ces traits, il est possible de recomposer une logique modale [Mas94]. Cette technique de composition a cependant deux inconvénients. Tout d'abord, la liste des traits proposés est finie et, par conséquent, le nombre de logiques accessibles est borné. De plus, seul un certain nombre de combinaisons est « autorisé » et, si l'on choisit des traits aléatoirement hors de ces combinaisons autorisées, aucune garantie de complétude ou de cohérence n'est fournie. Enfin, aucune technique n'est fournie pour déterminer automatiquement quelles combinaisons sont valides. À noter que ces conclusions s'appliquent également aux logiques de descriptions, du fait de leur forte similitude avec les logiques modales. Dans le cas des foncteurs logiques, toutes les compositions peuvent être formulées mais toutes ne sont pas valides ; la différence avec le schéma proposé précédemment est que le diagnostic est ici automatique et *a posteriori*, ce qui permet de composer des foncteurs logiques avec d'autres non définis lors de la création des premiers.

3.5.2 Avantages des foncteurs logiques

Les foncteurs logiques permettent en outre de manipuler des structures non complètes. Ainsi, la logique *Atom* n'est pas complète puisqu'elle ne prend en compte ni la conjonction ni la disjonction ; le treillis correspondant à la logique *Atom* n'est donc pas complet, mais rien n'empêche de manipuler cette logique. Les foncteurs logiques présentent ici un avantage par rapport à PAG, technique de génération automatique d'analyses à destination des compilateurs basée sur la composition de structures de données récursives. En effet, là où les foncteurs logiques autorisent la combinaison d'éléments non complets, PAG impose que les treillis manipulés soient complets [AM95].

En outre, les foncteurs logiques permettent de réintroduire les opérations manquantes en appliquant un nouveau foncteur logique ; dans le cas de *Atom*, le foncteur logique *Prop* est appliqué et on recrée ainsi la logique propositionnelle *Prop(Atom)*. Cependant, lors de la création d'un tel foncteur logique, un message de diagnostic automatique prévient alors le développeur que *Prop(Atom)* n'est pas complet vis-à-vis de la relation de subsomption \sqsubseteq ; il convient alors d'appliquer le foncteur logique *Set* et ainsi créer *Prop(Set(Atom))* qui assure la complétude. De plus, les possibilités de combinaisons des foncteurs logiques ne sont pas bornées, puisqu'aucune limite n'est fixée sur la taille des termes ainsi créés. En outre, les foncteurs logiques fonctionnent en système ouvert puisqu'un foncteur logique *Prop(X)* peut être utilisé pour un *X* défini après la création de *Prop*.

Enfin, les systèmes d'information logiques sont fortement génériques, puisqu'indépendants de la logique ; il suffit en effet de changer le *plugin* logique. Précisons néanmoins que seules des variations des logiques propositionnelles ont pour l'heure été implémentées. En outre, la création d'un système d'information logique n'est *a priori* pas accessible à un développeur d'applications sans connaissance en logique. Cependant, en combinant la technique des foncteurs logiques avec les systèmes d'information logiques, ce problème est résolu puisque le développeur combinera les foncteurs logiques de son choix afin de créer la logique adéquate utilisable dans son système. Un non-logicien est parfaitement à même de créer ainsi sa propre logique, comme l'a montré Olivier Bedel, doctorant dans l'équipe LIS, qui a créé une logique de coordonnées géographiques grâce à la composition suivante de foncteurs logiques :

$$Prod(Interval(Float), Interval(Float))$$

On peut ainsi représenter des coordonnées telles que (3.12, 7.07) ou des surfaces telles que (2..6, 5..10) et vérifier, par exemple, que ces coordonnées sont bien incluses dans cette surface.

4 Conclusion

Cette étude bibliographique a analysé deux types de systèmes d'information permettant l'indexation et la recherche de documents — en l'occurrence le Web sémantique et les systèmes d'information logiques — et les a décrits et comparés au regard de critères d'expressivité, de connaissances du domaine, et de généralité. Le Web sémantique connaît une forte expressivité, en particulier grâce aux logiques de descriptions, utilisables afin de poser des requêtes sur des

documents XML. Par ailleurs, les systèmes d'information logiques sont particulièrement génériques au niveau logique, laissant ainsi une grande liberté au développeur pour définir sa propre logique en fonction des besoins.

Il serait alors souhaitable de pouvoir recomposer les logiques de descriptions par les foncteurs logiques, afin d'augmenter les possibilités d'applications et d'atteindre l'expressivité de ces logiques, notamment à destination du Web sémantique. En outre, afin de mieux séparer les rôles lors du développement d'une application, il serait souhaitable de pouvoir découpler la notion de terminologie des foncteurs logiques, de manière à ce que de simples développeurs d'applications puissent sélectionner la terminologie qu'ils souhaitent utiliser pour ensuite concevoir leur système à partir des foncteurs logiques existants, et ce sans avoir à définir des foncteurs logiques ou des terminologies.

Remerciements

Merci à Mireille Ducassé et Olivier Ridoux pour leurs remarques constructives, et tout particulièrement à Sébastien Ferré pour avoir suivi de près la rédaction de cette étude bibliographique.

Références

- [AM95] Martin Alt and Florian Martin. Generation of efficient interprocedural analyzers with PAG. In *Static Analysis Symposium*, pages 33–50, 1995.
- [BCD⁺03] D. Berardi, D. Calvanese, G. DeGiacomo, M. Lenzerini, and M. Mecella. *e-service composition by description logics based reasoning*, 2003.
- [BFGHK05] Peter Baumgartner, Ulrich Furbach, Margret Gross-Hardt, and Thomas Kleemann. Optimizing the evaluation of XPath using description logics. In *Applications of Declarative Programming and Knowledge Management : 18th Workshop on Logic Programming, WLP 2004*, volume 3392 of *Lecture Notes in Artificial Intelligence*, pages 1–15. Springer, 2005.
- [BFRQ06] O. Bedel, S. Ferré, O. Ridoux, and E. Quesseveur. GEOLIS : A logical information system for geographical data. In *Int. Conf. Spatial Analysis and GEomatics – SAGEO 2006*, 2006.
- [BPSM⁺06] Tim Bray, Jean Paoli, C. Michael Sperberg-McQueen, Eve Maler, François Yergeau, and John Cowan. Extensible markup language (XML) 1.1 (second edition). World Wide Web Consortium, 2006. <http://www.w3.org/TR/2006/REC-xml11-20060816>.
- [CGL99] Diego Calvanese, Giuseppe De Giacomo, and Maurizio Lenzerini. Representing and reasoning on XML documents : A description logic approach. *Journal of Logic and Computation*, 9(3) :295–318, 1999.
- [DLNN97] F. M. Donini, M. Lenzerini, D. Nardi, and W. Nutt. The complexity of concept languages. *Information and Computation*, 134(1) :1–58, 1997.

- [Fer02] S. Ferré. *Systèmes d'information logiques : un paradigme logico-contextuel pour interroger, naviguer et apprendre*. Thèse d'université, Université de Rennes 1, 2002.
- [FR02] S. Ferré and O. Ridoux. A framework for developing embeddable customized logics. In A. Pettorossi, editor, *Int. Work. Logic-based Program Synthesis and Transformation*, LNCS 2372, pages 191–215. Springer, 2002.
- [FR04] S. Ferré and O. Ridoux. An introduction to logical information systems. *Information Processing & Management*, 40(3) :383–419, 2004.
- [FR06] S. Ferré and O. Ridoux. Logic functors : A toolbox of components for building customized and embeddable logics. Research report, Irisa, 2006.
- [GW99] Bernhard Ganter and Rudolf Wille. *Formal Concept Analysis : Mathematical Foundations*. Springer Verlag, 1999.
- [HS05] Ian Horrocks and Ulrike Sattler. A tableaux decision procedure for *SHOIQ*. In *IJCAI*, pages 448–453, 2005.
- [KM06] Yevgeny Kazakov and Boris Motik. A resolution-based decision procedure for *SHOIQ*. In *IJCAR*, pages 662–677, 2006.
- [LHRJ99] Arnaud Le Hors, Dave Raggett, and Ian Jacobs. HTML 4.01 Specification, 1999.
<http://www.w3.org/TR/1999/REC-html401-19991224>.
- [Mas94] Fabio Massacci. Strongly analytic tableaux for normal modal logics. In Alan Bundy, editor, *Proceedings of the Twelfth International Conference on Automated Deduction (CADE'94)*, volume 814, pages 723–737, Berlin, 1994. Springer-Verlag.
- [MBE03] Brahim Medjahed, Athman Bouguettaya, and Ahmed K. Elmagarmid. Composing web services on the semantic web. *The VLDB Journal*, 12(4) :333–351, 2003.
- [Nap97] Amedeo Napoli. Une introduction aux logiques de descriptions. Research report, Inria, 1997.
- [PR03] Yoann Padioleau and Olivier Ridoux. A logic file system. In *USENIX Annual Technical Conference, General Track*, pages 99–112, 2003.
- [Rij87] C J Van Rijsbergen. A new theoretical framework for information retrieval. *SIGIR Forum*, 21(1-2) :23–29, 1987.
- [Sch91] Klaus Schild. A correspondence theory for terminological logics : preliminary report. In *Proceedings of IJCAI-91, 12th International Joint Conference on Artificial Intelligence*, pages 466–471, Sidney, AU, 1991.
- [WS01] W3C Semantic Web Activity, 2001. <http://www.w3.org/2001/sw/>.
- [XPa99] XML Path Language (XPath) Version 1.0, W3C Recommendation, 1999. <http://www.w3c.org/TR/xpath>.