Découverte d'associations sémantiques pour le Web Sémantique Géospatial - le framework ONTOAST

Alina Dia Miron

UNIVERSITÉ JOSEPH FOURIER

N° attribué par la bibliothèque

|_|_|_|_|_|_|_|_|_|_|_|

THÈSE pour obtenir le grade de

DOCTEUR DE L'UNIVERSITE JOSEPH FOURIER

Spécialité : « Informatique »

préparée au laboratoire d'Informatique de Grenoble (LIG) dans le cadre de **l'École Doctorale « Mathématiques, Informatique, Sciences et Technologie de l'Information »**
et soutenue publiquement par

Alina Dia MIRON

le 8 décembre 2009

Titre :

Découverte d'associations sémantiques pour le Web Sémantique Géospatial: le *framework* ONTOAST

---

sous la direction de Jérôme GENSEL et de Marlène VILLANOVA-OLIVER

JURY

Mme. Florence LE BER	Rapporteur
Mme. Anne RUAS	Rapporteur
Mme. Marie Christine ROUSSET	Examinateur
Mme. Frédérique SEGOND	Examinateur
M. Alain BOUJU	Examinateur
M. Jérôme GENSEL	Examinateur
Mme. Marlène VILLANOVA-OLIVER	Examinateur

"La solitude est utile. Il faut parfois ne parler qu'avec soi-même. On entend alors de dures vérités ou d'agréables mensonges selon qu'on s'analyse ou qu'on s'imagine."

Henri de Régnier

Remerciements

Je tiens à remercier :

Madame Marie Christine ROUSSET, Professeur à l'Université Joseph Fourier de Grenoble, qui a accepté de présider le jury et d'examiner mon travail.

Madame Florence LE BER, Ingénieur en chef des Ponts, des Eaux et des Forêts, à l'Ecole Nationale du Génie de l'Eau et de l'Environnement de Strasbourg et Madame Anne RUAS, Ingénieur des Ponts et Chaussées à l'Institut Géographique National, qui m'ont fait l'honneur de rapporter mon travail et dont les remarques m'ont permis d'en améliorer certains aspects.

Merci également aux autres membres du jury qui ont accepté de juger ce travail : Madame Frédérique SEGOND, Chercheur à Xerox's European Research Center et Monsieur Alain BOUJU, Maître de Conférences à l'Université de la Rochelle.

Mes plus sincères remerciements à Monsieur Jérôme GENSEL, Professeur à l'Université Pierre Mendès France, mon directeur de thèse, pour son encadrement et ses nombreux conseils. La confiance que tu m'a accordée m'a permis de m'épanouir en tant que jeune chercheur et de mener à bon port cette thèse. Je te remercie également pour ton soutien pendant les périodes plus difficiles, quand nos travaux ont été remis en question. Tu m'as appris à défendre mes idées avec passion mais aussi à chercher sans cesse à les améliorer. Ta rigueur, ton énergie et la passion que tu investis dans ton travail ont été (et resteront) une vraie inspiration pour moi. Encore une fois, merci pour tout !

Je voudrais également remercier Madame Marlène VILLANOVA-OLIVER, Maître de Conférences à l'Université Pierre Mendès France, ma co-encadrante, pour toute sa coopération et toutes nos discussions qui ont beaucoup contribué à l'avancement de mon travail de thèse. Je te remercie pour ta patience, pour ton soutien continuu, pour ton aide (surtout pendant la rédaction et la correction de mon mémoire) et pour ton intérêt dans mon travail. J'ai beaucoup apprécié tes encouragements et tes conseils. J'ai trouvé en toi une vraie amie et je t'en remercie !

Je remercie également à tous ceux qui m'ont soutenue et encouragée tout au long de ces trois années de thèse. J'ai une pensée spéciale pour les permanents de l'équipe STEAMER :

Monsieur Hervé MARTIN, Professeur à l'Université Joseph Fourier, merci de m'avoir accueilli dans l'équipe STEAMER. Merci aussi pour votre soutien et pour votre confiance en moi et en mes travaux.

Paule-Annick DAVOINE, Maître de Conférences à l'INP de Grenoble pour sa gentillesse et son amitié. Merci pour ton soutien, pour ta patience et pour m'avoir appris le vocabulaire des

géographes.

Danielle ZIEBELIN et Philippe GENOUD pour votre disponibilité et votre soutien ainsi que pour nos discussions sur AROM.

Mes collègues d'équipe : José, Marius, Celine, Bogdan, Raphaël, Laurent, Windson, Christine, Carlos, Angela, Aurélie, Sandro.

Mes amis roumains et grenoblois que je n'ai pas mentionnés ci-dessus mais dont la compagnie et le soutien n'ont pas été les moindres.

Bien évidemment, je remercie mes parents et ma famille, qui ont toujours cru en moi et qui m'ont toujours soutenue. Merci papa, sans toi je n'aurais jamais pu goûter aux joies de l'informatique ! Merci maman, sans toi je ne serais jamais partie dans cette aventure française ! Malgré les 2000 km qui nous ont séparés pendant les 5 dernières années vous avez su être toujours près de moi pour m'encourager dans les moments difficiles et pour fêter mes réussites.

Grand merci enfin à Petre pour m'avoir convaincue de relever ce défi et pour tous les bons moments que nous avons passé ensemble ces trois derniers années.

Dia

TABLE DES MATIÈRES

TABLE DES FIGURES

INTRODUCTION

Chapitre 1

INTRODUCTION

1.1 Contexte du travail

Le contexte général dans lequel s'inscrit ce travail de thèse est celui du Web Sémantique [34] dont l'ambition est d'associer à chaque ressource du Web une représentation formalisée de son contenu, afin de permettre une exploitation qualitativement supérieure de grandes quantités de données et de services divers. Le Web Sémantique entend donc proposer des solutions pour accroître la précision et le rappel des moteurs de recherche, à travers l'annotation du contenu des ressources Web. Cette annotation est mise en œuvre à l'aide de *concepts* et de *relations ontologiques*, compréhensibles et exploitables à la fois par l'homme et les machines. En associant une couche descriptive aux pages Web classiques, le Web Sémantique rend possible l'évolution des données vers des connaissances et marque le début d'une nouvelle étape de développement de l'Internet.

Dans ce contexte, l'information spatiale et temporelle occupe une place particulière. Notamment parce que, de nos jours, l'essor des technologies mobiles (en termes de capacité des dispositifs et des réseaux) permet à un utilisateur nomade d'accéder à des informations depuis - théoriquement - n'importe où. L'ajout progressif de capteurs (GPS, boussole, lecteur RFID...) aux dispositifs mobiles conduit désormais à l'émergence de systèmes mobiles capables de capturer automatiquement des données contextuelles et d'adapter les informations délivrées à un utilisateur en fonction de ce contexte d'utilisation (*i.e.* les caractéristiques matérielles du dispositif, la *localisation* de l'utilisateur, le *moment* de la prise de vue, *etc.*) [234].

Cette situation est très bien décrite par Bradley Horowitz : "*Mon téléphone sait toujours l'heure qu'il est. Il sait approximativement toujours où je suis via le GPS ou via le réseau téléphonique qu'il utilise. Si le système sait aussi que je suis présent à tel évènement à telle heure (via mon agenda ou mes messages), alors quand je prends une photo, le système est capable d'automatiser l'étiquetage de cet évènement et d'introduire les méta-données automatiquement. C'est ce vers quoi nous tendons : un monde où le qui, quoi, où et quand peuvent être générés, lus et résolus automatiquement par les machines.*" [188].

Cependant, plusieurs défis scientifiques et technologiques doivent être relevés afin de permettre le développement de ces nouveaux systèmes intelligents. Parmi ces défis, on trouve en premier lieu le développement de techniques de représentation de l'information spatiale et temporelle qui soient compatibles avec les langages du Web Sémantique afin de soutenir une intéroperabilité maximale, ainsi que l'exploitation automatique de ces informations par les agents logiciels (*i.e.* moteurs de recherche sémantiques, gestionnaires de collections de photos géolocalisées, guides touristiques, *etc.*).

Des moteurs de recherche intelligents pourraient exploiter ces annotations et les combiner avec des informations provenant d'autres sources de données spatiales et temporelles du Web afin d'offrir des moyens étendus de recherche. Or, si on constate que, dans la plupart des cas, les com-

posantes spatiales et temporelles ne sont pas systématiquement utilisées (notamment lors de la recherche de documents via les moteurs existants), elles peuvent s'avérer utiles, par exemple, pour préciser la formulation de certaines requêtes. Ainsi, dans le cas de la recherche d'une personne à partir de son nom, l'emploi d'un moteur de recherche sur ce seul critère débouche souvent sur plusieurs résultats, qui ne font pas tous référence au même individu. Une solution pour réduire le nombre de réponses possibles, serait d'utiliser les informations concernant la localisation de la personne visée (sa ville de résidence, l'adresse de son travail, *etc.*). Pour autant, l'exploitation d'une information géospatiale, qu'elle soit une donnée ou une méta-donnée, requiert des formalismes de représentation et de raisonnement adaptés aux particularités de ce type d'information et compatibles avec le Web Sémantique [34].

En vue de répondre aux besoins spécifiques de l'information spatiale et temporelle dans le cadre du Web Sémantique, Egenhofer a proposé l'idée d'un Web Sémantique Géospatial [88]. Celui-ci regrouperait les activités autour du développement d'*ontologies spatiales et temporelles* pour la description sémantique d'informations à références spatiales et temporelles accessibles sur le Web. L'objectif du Web Sémantique Géospatial est semblable à celui du Web Sémantique : associer aux ressources Web des descriptions (méta-données) ici, spatio-temporelles, interprétables par les humains, et surtout par les machines, afin que le traitement automatisé de ces données par des agents logiciels soit rendu plus performant.

1.2 Problématique

Information spatiale et temporelle sur le Web Sémantique Géospatial

Deux défis principaux doivent être relevés pour permettre l'exploitation des informations spatiales et temporelles dans le cadre du Web Sémantique Géospatial. Dans un premier temps, ces informations doivent être décrites à l'aide d'un formalisme déclaratif (langage ou ontologie) qui favorise l'interopérabilité. Plusieurs langages à ontologies ont été définis jusqu'à présent avec l'ambition d'appuyer la construction du Web Sémantique, notamment RDF [144] et RDF Schema [44], OWL [165] et OWL 2 [104]. Ces langages, très génériques, permettent de répondre avec succès à la plupart des exigences d'expressivité imposées par les divers domaines d'application abordés. Néanmoins, vis-à-vis de la représentation de l'*information spatiale*, on peut constater l'absence de structures dédiées (syntaxiques, typées, *etc.*). Il convient donc de gérer les informations spatiales à l'aide des concepts et relations spatiaux définis dans des ontologies spécialisées.

À ces fins, des nombreux travaux proposent des ontologies pour la description des éléments (ou *features*) spatiaux, à travers l'utilisation de concepts spatiaux qui remplacent/simulent les types spatiaux absents de la définition du langage OWL [165]. Notamment, l'ontologie *GeoRSS-Simple* [151] est devenue recommandation du groupe de travail Geospatial Incubator Group (GeoXG) dans le domaine de la représentation des données spatiales quantitatives pour le Web Sémantique. Néanmoins, à ce jour, les différents acteurs du Web Sémantique Géospatial ne sont pas parvenus à un consensus en ce qui concerne la définition d'une ontologie de relations spatiales qualitatives qui puisse être recommandée comme standard d'annotation.

Cependant, une telle ontologie est nécessaire pour pouvoir exprimer des faits tels que "*cette photo a été prise très près de la tour Eiffel*", "*mon correspondant téléphonique se trouve à l'intérieur du centre commercial*", "*le camping se trouve au sud de Marseille*", *etc.* Ces types de relations sont utiles dans le processus d'annotation, lorsqu'on ne dispose pas de données géomé-

triques exactes. Elles sont également plus intuitives et faciles à appréhender car adaptées à notre perception de l'espace et du temps. De plus, les relations spatiales qualitatives peuvent être exploitées pour la déduction de relations implicites, moyennant un ensemble de règles d'inférence. Elles constituent donc une alternative de représentation très intéressante dans le cas ou les données quantitatives sont manquantes, et permettent d'opérer des déductions lorsque les connaissances sur les objets utilisés ne sont pas complètes ou lorsque les détails et la précision sont secondaires.

Du côté des informations temporelles pour le Web Sémantique, on dispose en revanche d'un ensemble de types temporels défini par le Schéma XML et repris par le langage OWL. Cet ensemble est complété par l'ontologie *OWL Time* [132], récemment proposée par le W3C comme référence pour la description du contenu temporel des pages Web ou pour la description des propriétés temporelles des services Web.

Dans un deuxième temps, si plusieurs approches pour la modélisation des informations spatiales et temporelles pour le Web Sémantique existent à ce jour, il faut cependant souligner l'absence de moteurs d'inférence spatiaux et temporels capables d'exploiter ces données pour déduire des informations implicites ou/et pour répondre à des requêtes. De fait, les annotations spatiales et temporelles restent très peu exploitées par les moteurs de recherche, alors qu'elles pourraient apporter plus de précision, affiner la requête, limiter la recherche en utilisant des contextes temporels et spatiaux, *etc.*

Dans cette optique, nous nous intéressons à l'exploitation d'informations spatiales et temporelles quantitatives et qualitatives, afin de répondre à des requêtes du type :

- quelles sont les photos qui ont été prises **au nord** de la tour Eiffel ?
- quels sont les musées **proches** de ma position actuelle ?
- quels sont les spectacles qui auront lieu **pendant** le mois d'Octobre ? ...

Afin de proposer une exploitation adaptée des informations spatiales et temporelles, il faut définir des outils de raisonnement spatial et temporel, capables d'exploiter les informations définies en utilisant à la fois les types temporels proposés par les langages d'ontologies (OWL et OWL 2) et les ontologies spatiales et temporelles standard. Parmi les fonctionnalités qui doivent être mises à disposition par ces moteurs d'inférence priment :

a) la capacité d'interpréter une requête spatio-temporelle qualitative définie par rapport à une ontologie spatiale et temporelle de référence ;

b) la vérification de l'existence, dans la base ontologique interrogée, de relations particulières et, dans le cas où elles ne sont pas explicites, la preuve de leur existence, ou, à l'inverse, la preuve de leur non existence, en exploitant des informations spatiales et temporelles quantitatives et des relations spatiales et temporelles qualitatives.

Analyse sémantique pour le Web Sémantique Géospatial

Comme c'est le cas pour le Web Sémantique, la plupart des travaux autour du Web Sémantique Géospatial portent sur l'intégration, l'interopérabilité et la recherche d'information à l'aide d'ontologies. Récemment, est apparu un courant de recherche, nommé *analyse sémantique* (en anglais, Semantic Analytics) [219, 19], qui s'intéresse à l'analyse des entités, et, plus particulièrement à l'élicitation de leurs relations. Dans l'hypothèse où A et B sont deux ressources décrites à travers des graphes RDF(S), ces travaux visent à répondre à la question : *comment A est-elle reliée à B* ? Concrètement, l'objectif de l'*analyse sémantique* est de mettre en exergue des *associations sémantiques* [219, 19, 10] qui sont des associations complexes (au sens où elles peuvent impliquer

plusieurs autres associations) liant deux ressources dans un graphe RDF(S), en s'appuyant sur une mesure de connectivité (existe-t-il un chemin de A à B ?) et une mesure de similarité (le graphe contenant A est-il semblable au graphe contenant B ?). Ce nouveau domaine de recherche a de nombreuses applications potentielles : analyse de texte, bioinformatique, prévention des risques naturels, éducation, sécurité, *etc.*

Néanmoins, à ce jour, l'analyse sémantique présente plusieurs limitations induites par l'expressivité limitée des graphes RDF(S) qu'elle exploite. En conséquence, les associations sémantiques liant deux individus ont une portée limitée, ne pouvant pas dépasser le cadre du graphe RDF(S) courant.

Dans ce sens, il serait intéressant d'étudier l'adaptation de l'analyse sémantique pour des ontologies OWL 2 DL, plus expressives que les graphes RDF(S), et offrant une palette plus large d'axiomes ou d'assertions possibles. Une telle analyse pourrait découvrir davantage d'associations sémantiques, en s'appuyant sur l'inférence de relations implicites à l'aide des raisonneurs évolués disponibles pour OWL DL, tels que Pellet ou RacerPro.

Compte tenu de la possible taille conséquente des ontologies explorées et du grand nombre potentiel d'associations sémantiques établies, un enjeu important pour l'analyse sémantique est de ne retenir que les plus pertinentes au regard de critères exprimés [10, 11]. À ces fins, la notion de *contexte d'une requête* est introduite par [219, 19], comme l'ensemble de classes et de propriétés jugées prioritaires par rapport à la requête courante. C'est pourquoi, il nous apparait opportun de donner la possibilité à l'utilisateur de spécifier de tels contextes.

Jusque là, les techniques d'*analyse sémantique* ont opéré majoritairement sur la dimension thématique (donc générique) de l'information. Néanmoins, Perry *et al.* ont proposé d'œuvrer pour une *analyse sémantique géospatiale* [200, 199]. Celle-ci repose sur la description d'ontologies RDF(S) qui capturent les dimensions spatiale et temporelle de l'information. La notion de *contexte spatial et temporel* définie ici limite la recherche à l'ensemble des ressources liées par des relations valides durant pendant un intervalle de temps et pour une région géographique fixés. Il s'agit donc d'une façon de circonscrire la recherche d'*associations sémantiques* aux limites définies par un contexte spatial et temporel donné.

L'approche de Perry *et al.* exploite exclusivement des informations spatiales et temporelles quantitatives pour la validation d'une association sémantique par rapport à un contexte donné, ce qui en réduit considérablement la portée.

Au-delà de la sélection d'*associations sémantiques*, la prise en compte des dimensions spatiales et temporelles, dans les limites d'un contexte spatio-temporel, peut également faire émerger de nouvelles *associations sémantiques* entre des ressources. Par exemple, pour deux personnes qui travaillent *en même temps, au même endroit*, une association sémantique témoignant de cette proximité peut suggérer que les deux personnes *se connaissent* probablement, selon la granularité de l'information temporelle et spatiale considérée.

1.3 Aperçu de la proposition

Afin de répondre aux problématiques décrites ci-dessus, nous proposons un raisonneur spatial et temporel, appelé ONTOAST, qui peut être utilisé pour exploiter les informations décrites dans les ontologies du Web Sémantique Géospatial, grâce à sa compatibilité avec le langage OWL 2 DL. Construit comme extension du système de Représentation des Connaissances par Objets AROM

[193], ONTOAST est à la fois un environnement de modélisation d'ontologies spatio-temporelles et un système d'inférence, capable de raisonner sur des connaissances spatiales, temporelles et thématiques.

Le système ONTOAST est chargé d'interpréter des descriptions spatiales et temporelles quantitatives et qualitatives réalisées conformément à des ontologies telles que *GeoRSS-Simple* ou *OWL Time*. Ces descriptions sont traduites vers un formalisme propre à ONTOAST afin qu'elles puissent être exploitées par des mécanismes de raisonnement. Après une étape d'inférence, les relations spatiales et temporelles résultantes sont traduites dans OWL 2. Les connaissances ainsi explicitées sont mises à la disposition des raisonneurs du monde des ontologies.

ONTOAST est muni d'un ensemble de relations spatiales et temporelles qualitatives prédéfinies, qui accroissent sa flexibilité et son expressivité, et permettent des raisonnements lorsqu'on ne dispose pas de données spatiales précises. Il s'agit de relations spatiales *topologiques*, d'*orientation*, et de *distance*, et de relations de *topologie temporelle*, déclarées explicitement par l'utilisateur, ou bien inférées à partir des informations existantes.

Lors d'une requête à propos de la relation spatiale/temporelle qualitative qui existe entre deux régions de l'espace/intervalles de temps, la réponse délivrée par ONTOAST sera construite :

- en utilisant les connaissances explicites, dans le cas où la relation cherchée est stockée dans la base ;
- en réalisant des inférences qualitatives à partir des relations explicites qui existent entre les objets ;
- en utilisant des méthodes numériques d'estimation et de calcul pour déduire des relations qualitatives impliquant les individus visés par la requête.

La deuxième partie notre proposition vise l'utilisation du raisonneur ONTOAST pour la découverte d'associations sémantiques entre les individus décrits dans des ontologies OWL 2 DL. Notre approche a deux caractéristiques principales :

a) adaptée aux ontologies OWL 2 DL, l'analyse sémantique peut donc exploiter les axiomes de ce langage, accédant ainsi à une palette étendue d'inférences, qui augmentent le nombre d'associations sémantiques qu'il est possible de découvrir. De plus, en bénéficiant des axiomes d'alignement de OWL 2 (*i.e.* `EquivalentClasses`, `EquivalentProperties`, `SameIndividual`, *etc.*), la découverte d'associations sémantiques qui recouvrent plusieurs ontologies devient possible.

b) les capacités d'inférence offertes par ONTOAST peuvent être utilisées afin de limiter l'espace de recherche, et pour rendre explicites les relations spatiales et temporelles qualitatives implicites. Celles-ci peuvent alors être utilisées pour la déduction de nouvelles associations sémantiques.

Notre proposition consiste en un *framework* pour la découverte des associations sémantiques entre des individus décrits dans des ontologies OWL DL et/ou OWL 2 DL. Celui-ci est centré sur l'utilisation du système ONTOAST comme raisonneur spatial et temporel.

Afin d'exploiter les raisonnements fournis par ONTOAST, les connaissances ontologiques doivent être traduites vers le formalisme d'ONTOAST et stockées dans un entrepôt de connaissances orientées-objets. Pour interroger l'entrepôt, l'utilisateur spécifie sa requête dans la forme d'une paire d'individus (x,y) en utilisant une *Interface d'interrogation*, qui permet aussi la spécification des *Contextes Spatial, Temporel* et *Thématique* attachés à la requête. Afin de réduire l'es-

pace de recherche, le processus de découverte des associations sémantiques débute par le filtrage des connaissances disponibles, en suivant les spécifications des trois contextes. Les connaissances filtrées sont exploitées par un module de *Découverte des associations sémantiques*. Le Raisonneur ONTOAST est utilisé à la fois dans la phase de filtrage (pour déduire des caractéristiques géographiques des individus ainsi que la validité temporelle des individus et des tuples), et dans la phase de découverte des associations sémantiques (pour déduire les connexions spatiales et temporelles entre individus). Les associations sémantiques découvertes sont ensuite classées par un module de *Classification de résultats*, en utilisant les spécifications de contexte et présentés à l'utilisateur par un module de *Visualisation des résultats*.

1.4 Plan

La première partie de la thèse, consacrée à l'état de l'art est composée de trois chapitres :

- le chapitre 2, présente l'état d'avancement des travaux sur la représentation et le raisonnement à base d'informations spatiales et temporelles. Nous décrivons les principales théories spatiales et temporelles qui ont été développées et implémentées, dans des domaines tels que les *Systèmes d'Information Géographique*, la *représentation et le raisonnement qualitatifs*, les *bases de données*, *etc.*
- le chapitre 3 présente les travaux que nous jugeons les plus importants relativement à la représentation spatiale et temporelle à l'aide d'ontologies OWL. Nous partons d'une étude générale sur le Web Sémantique, qui vise les langages proposés pour la représentation des ontologies, les outils de modélisation et les outils de raisonnement qui existent à ce jour. Ensuite, nous étudions ces langages et outils du point de vue de l'information spatiale et temporelle, et nous présentons les principaux défis pour la mise en place d'un Web Sémantique Géospatial.
- le dernier chapitre de l'état de l'art, le chapitre 4, propose un tour d'horizon des techniques d'exploitation du Web et du Web Sémantique qui existent à ce jour. Il décrit également un nouveau paradigme de recherche, l'analyse sémantique, dont le but est d'exploiter les ontologies du Web Sémantique pour découvrir les connexions qui existent entre les individus.

Dans la deuxième partie de cette thèse nous présentons nos propositions :

- le chapitre 5 décrit la construction d'un système hybride, AROM-ONTO, pour la représentation et le raisonnement à base d'ontologies, inspiré à la fois de la Représentation des Connaissances par Objets et des Logiques de Description.
- le chapitre 6 décrit une extension du système AROM-ONTO, appelée ONTOAST, qui englobe un modèle spatial et temporel prédéfini. ONTOAST propose plusieurs algorithmes de raisonnement spatial et temporel, et peut être utilisé dans le contexte du Web Sémantique Géospatial pour exploiter les annotations spatiales et temporelles décrites en OWL DL ou OWL 2 DL. Il intègre les types temporels proposés par ces deux langages et/ou les concepts spatiaux et temporels définis par les ontologies *GeoRSS-Simple*, *OWL-Time* et *QualitativeSpatialRelations*. L'ontologie *QualitativeSpatialRelations* est l'ontologie que nous proposons pour la représentation des relations spatiales qualitatives.
- le chapitre 7 est consacré à l'adaptation de l'analyse sémantique pour les ontologies OWL DL et OWL 2 DL. Il présente également un *framework* pour l'analyse sémantique géospatiale, basé sur l'utilisation d'ONTOAST comme raisonneur spatial et temporel pour la

déduction des connexions spatiales et temporelles implicites existant entre les individus décrits dans des ontologies OWL DL ou OWL 2 DL.

– enfin, le dernier chapitre dresse le bilan de cette thèse et présente les perspectives possibles.

Première partie

ETAT DE L’ART

Chapitre 2

Représentation quantitative et qualitative de l'espace et du temps

Le temps et l'espace sont infinis, et pourtant on n'en a jamais assez.

C. Thomson

SOMMAIRE

Résumé

*Ce chapitre présente l'état d'avancement des recherches sur la **représentation** et le **raisonnement** consacrés au le temps et à l'espace. De nombreuses théories spatiales et temporelles ont été développées et implémentées dans des domaines tels que les **Systèmes d'Information Géographique**, la **représentation** et le **raisonnement qualitatifs**, les **bases de données**, etc.*

*De ces travaux, la construction du **Web Sémantique Géospatial** doit s'inspirer afin d'offrir non seulement la possibilité d'annoter les ressources du Web avec des informations spatiales et temporelles issues d'ontologies, mais également les capacités nécessaires pour exploiter ces données. L'exploitation de ces données spatiales et temporelles vise notamment l'élicitation d'informations implicites, la traduction d'informations quantitatives vers des relations qualitatives plus compréhensibles pour les humains, une recherche localisée, etc.*

2.1 Introduction

L'espace et le temps sont des notions de base dans la communication et la vie de tous les jours [30]. Ils sont des concepts fondamentaux dans la géographie et ont un rôle clé dans la plupart des domaines de l'activité humaine [201]. Malgré cela, un consensus général n'a pas encore été établi sur la sémantique à associer aux concepts et aux relations spatiaux et temporels. Par exemple, la relation `NordOf` a des sémantiques différentes en fonction de l'approche de modélisation utilisée PDR [107], CDR [224], en fonction des types d'objets spatiaux pris en compte (des régions ou des points), en fonction du système de référence, *etc.*

Ainsi, la représentation de l'espace et du temps est un sujet de recherche actif depuis plus de vingt ans. Le thème est interdisciplinaire et suscite l'intérêt de différentes communautés scientifiques, non seulement en informatique, mais aussi en géographie [36, 25, 53, 27, 99], en linguistique [146, 17], philosophie [52], et psychologie [137, 164]. En informatique, différents domaines, tels que les bases de données spatiales [121] et temporelles [81, 115], les systèmes d'information géographique (SIG) [89], les bases de données d'images [39, 35], le raisonnement spatial et temporel qualitatif [97, 67, 68] nécessitent des recherches sur la modélisation et le raisonnement à base de temps et de l'espace.

Dans ce chapitre, nous proposons un état de l'art des domaines de la représentation et du raisonnement à base d'informations spatiales (section 2.3) et temporelles (section 2.4) quantitatives et qualitatives. En effet, nous utilisons une partie de ces relations, ainsi que les méthodes de calcul et de raisonnement qui leur sont spécifiques, pour la définition d'un raisonneur spatial capable d'exploiter à la fois les informations spatiales quantitatives et les relations spatiales qualitatives afin de mettre en exergue des relations spatiales implicites (voir chapitre 6).

2.2 Différentes représentations de l'espace et du temps

2.2.1 Différents niveaux de représentation

Selon Clementini et Laurini [62], l'information spatiale peut être classée par catégories en fonction de trois niveaux différents de représentation : le *niveau géométrique*, le *niveau informatique*, et le *niveau utilisateur*.

Au *niveau géometrique*, les objets spatiaux peuvent être considérés comme des ensembles de points et/ou des régions, et les relations spatiales peuvent être formellement définies en termes mathématiques : par exemple, dans le modèle des neuf intersections proposé par [85], les relations topologiques sont définies sur les valeurs vides et non-vides des intersections de la frontière, de l'intérieur et de l'extérieur des deux objets (voir section 2.3.2.1). Le niveau géométrique peut être considéré comme le niveau le plus primitif pour l'étude des relations spatiales, puisqu'il en donne des définitions formelles. Les deux autres niveaux se rapportent toujours à la définition d'une relation spatiale au niveau géométrique.

Au *niveau informatique*, les objets spatiaux sont représentés comme des types de données spatiaux et les relations spatiales entre les objets sont, en général, calculées par des opérateurs spatiaux. Bien que le niveau géométrique puisse être considéré, par définition, sans erreur, la représentation des objets géographiques au niveau informatique est intrinsèquement concernée par l'approximation [62]. Dans les modèles raster (définis dans la section 2.3.1), l'approximation est liée à la taille des cellules, unités élémentaires de représentation, alors que dans les modèles vecto-

riels (définis dans la section 2.3.1) l'approximation est régie par le nombre de points représentatifs d'une courbe [62].

Au *niveau utilisateur*, les concepts spatiaux peuvent être définis à l'aide d'ontologies spatiales, comme illustré dans le chapitre 3. Les relations spatiales dépendent fortement, à ce niveau, de divers facteurs, tels que la particularité du domaine, l'imprécision et le flou implicite des termes employés par l'utilisateur, ou encore la variabilité des termes selon les différents pays et langages [62].

La catégorisation proposée par Clementini et Laurini [62] pour l'espace peut être adaptée pour l'identification des différents niveaux de représentation de l'information temporelle. On trouve ainsi, à la place du *niveau géométrique* le *niveau algébrique*, qui modélise les objets temporels comme des instants et/ou des intervalles de temps. Les relations temporelles s'appuient, à ce niveau, sur les résultats des comparaisons algébriques entre les instants de début et les instants de fin des intervalles. Au *niveau informatique*, les objets temporels sont modélisés à l'aide de types temporels, et les relations temporelles sont déduites à l'aide d'opérateurs temporels. Enfin, au *niveau utilisateur*, les concepts et les relations temporels sont définis à l'aide d'ontologies dédiées.

Pour illustrer l'interdépendance des trois niveaux de représentation de l'information spatiale, Clementini et Laurini [62] donnent l'exemple d'une application exécutée sur un dispositif mobile avec GPS et communication sans fil. Celle-ci donne des indications sur les bâtiments qui sont visibles par un utilisateur se déplaçant en voiture dans la ville. Cette situation peut conduire à formuler la requête suivante : « *quel est le nom des bâtiments situés* ***à droite*** *et* ***à gauche*** *de la position de l'utilisateur, pour lesquels il n'y a aucun autre obstacle entre ces bâtiments et l'utilisateur ?* ». Une autre requête qui viserait l'exploitation des informations temporelles est, par exemple : « *quels sont les événements qui doivent survenir aujourd'hui* ***pendant*** *l'intervalle de 15h jusqu'à 18h ou 2 heures* ***après*** *le dîner ?* »

Afin de pouvoir traiter des tels requêtes dans le contexte du Web Sémantique Géospatial, il faut mettre à disposition des outils capables de gérer la traduction continue des connaissances spatiales entre deux niveaux de représentation différents : le *niveau utilisateur* (présenté dans le chapitre 3) et le *niveau informatique* (dont traite ce chapitre), les deux formalisés par rapport au *niveau géométrique*/*algébrique*. La construction même du Web Sémantique Géospatial (présenté dans le chapitre 3) est basée sur l'annotation manuelle et/ou automatique des ressources Web avec des termes ontologiques qui relèvent du *niveau utilisateur*. Afin que les raisonneurs puissent exploiter ces annotations spatiales spécifiées à l'aide d'adresses, de coordonnées géographiques, de relations topologiques entre objets *etc.*, ils doivent traduire les descriptions dans des objets informatiques (*niveau informatique*) qui seront utilisés pour les calculs respectifs. Afin qu'ils soient compris par l'utilisateur, les résultats des calculs doivent être re-traduits en termes de relations ontologiques avant d'être présentés comme résultats de la requête.

2.2.2 Représentation quantitative *versus* représentation qualitative

La représentation qualitative offre des moyens pour modéliser des propriétés continues du monde réel en utilisant un *vocabulaire symbolique* limité. On parle alors de catégories *qualitatives* : "beau", " grand", "petit", "après-midi", "soir",... ou de relations qualitatives : "au Sud de", "plus grand que", " disjoint", " proche de" , ... Dans ce contexte, toutes les qualifications ne sont pas utiles, d'où le *principe de pertinence* évoqué par [94] qui impose que les distinctions faites par une qualification soient pertinentes pour le raisonnement visé.

Le raisonnement qualitatif permet d'opérer des déductions lorsque les connaissances sur les objets utilisés ne sont pas complètes, ou lorsque les détails sont secondaires. Ce type de raisonnement vise à diviser l'ensemble des valeurs possibles pour une variable donnée, en *classes d'équivalence* regroupant des valeurs proches qui seront considérées comme équivalentes. Classiquement, il existe une *relation d'ordre* (partielle ou totale) établie pour chaque espace symbolique, dont la *transitivité* peut être exploitée afin de réaliser des inférences. Une autre classe d'inférence est basée sur la construction des algèbres qualitatives, mais ces dernières peuvent produire des réponses ambigües [67, 65].

L'un des avantages majeurs de la représentation qualitative provient de la réduction de complexité qu'elle propose. En numérique, il faut travailler sur des ensembles de valeurs possibles, ce qui complexifie les calculs, alors que le qualitatif propose un nombre de distinctions adapté à la tâche visée, ainsi que la comparaison aisée entre valeurs inexactes [80]. Au contraire, l'utilisation de données numériques peut amener à faire plus de distinctions que nécessaire, et peut entraîner un coût de calcul plus important. De plus, le qualitatif peut également soutenir une meilleure précision, car il évite les problèmes de discrétisation et de perte d'information (*i.e.* l'égalité de deux régions A et B ne peut pas être validée dans un scénario quantitatif si les contours de A et B ne contiennent pas exactement les mêmes ensembles de points) [80].

2.3 Représentation de l'espace

Par la suite, nous présentons plusieurs approches de représentation spatiale qui relève du *niveau informatique*. Concrètement, nous étudions la représentation spatiale quantitative à l'aide de types de données et d'opérations dédiés, et la représentation spatiale qualitative, mise en œuvre à travers des algèbres de relations qualitatives. Ces théories sont spécifiées formellement en ayant recours au *niveau géométrique*. Le *niveau utilisateur* sera abordé dans le chapitre 3.

2.3.1 Représentation spatiale quantitative

En termes de représentation spatiale, les Systèmes d'Information Géographique (*SIG*) offrent une longue tradition d'intégration, de gestion et d'analyse efficaces des données spatiales quantitatives, modélisées dans un espace de coordonnées cartésien [217]. Pour pouvoir être utilisées, les *SIG* requièrent en général des descriptions précises de la localisation des objets géographiques qu'ils manipulent, définies en termes de contours géoréferencés modélisés à l'aide de types spatiaux tels que les *points*, les *lignes*, les *surfaces* et de leur inter-relations.

D'une manière générale, et selon [76], l'information géographique peut être définie comme une paire reliant une *information* relative à un *objet* ou un *phénomène* du monde terrestre (*i.e.* un bâtiment), et sa *localisation* sur la surface de la Terre, décrite dans un système de références explicite. Le premier élément de la paire, peut être qualifié de *thématique*, c'est-à-dire relatif à l'ensemble des critères descriptifs des objets, indépendants de leur localisation (*i.e.* le nombre d'étages, le nombre d'habitants, *etc.*). Le deuxième terme est relatif à la mesure de la position des objets sur la surface de la Terre, ainsi qu'à leurs formes et dimensions [76]. Certains SIG ajoutent à cette paire un troisième élément, le *temps*. Nous présentons les modalités pour modéliser la dimension temporelle de l'information dans la section 2.4.1.

Pour représenter la localisation, les SIG utilisent en général le modèle vectoriel ou le modèle raster. Le modèle *vectoriel* (ou modèle discret) propose la modélisation des objets à travers la

spécification de leurs contours en termes de *points*, de *lignes* ou de *polygones*, définis en coordonnées réelles (x,y). Chaque *ligne* est définie par une succession de points, et chaque surface (ou polygone) est définie par les arcs qui tracent ses limites [76, 157]. Ce modèle représente le monde comme un ensemble d'objets ayant des frontières très bien définies. Une illustration graphique d'une région de l'espace représentée en utilisant le modèle vectoriel est présentée dans la figure 2.1.

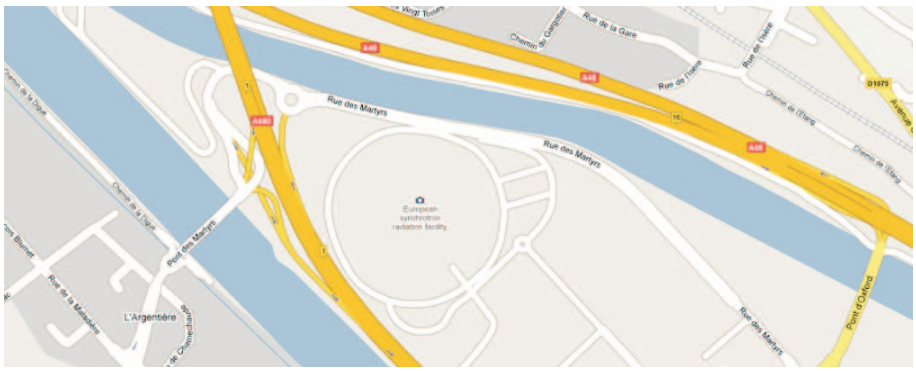


FIG. *2.1: Exemple de rivières, de routes et de quartiers modélisés en utilisant le modèle vectoriel.*

Le modèle *raster* (ou modèle continu) représente la surface de la région d'intérêt par une succession régulière de cellules, selon un maillage défini. À chaque cellule sont associées une ou plusieurs valeurs d'attributs qui caractérisent la cellule en question. En conséquence, le modèle continu représente le monde réel à travers un nombre fini de variables, chacune définie dans toutes les cellules considérées.

FIG. *2.2: Mêmes informations que celles présentées dans la figure 2.1, mais modélisées en utilisant le modèle raster.*

Le modèle raster est particulièrement utilisé comme fond de carte car il offre un rendu similaire aux cartes conventionnelles (voir figure 2.2) et a un grand impact visuel en réussissant à communiquer beaucoup d'informations [157]. L'avantage du modèle vectoriel est qu'il est plus adapté à l'abstraction et au raisonnement usuel.

Si on considère les types primitifs spatiaux proposés par les différents SIG pour modéliser l'information spatiale en suivant le paradigme vectoriel, on se rend compte qu'ils varient d'un outil à l'autre. Il en est de même pour les opérateurs et les fonctions proposés pour manipuler ces

types spatiaux de données [174]. Par exemple, le logiciel ArcInfo[1] met à disposition les types de données spatiaux suivants :

- les *points*, qui peuvent être des nœuds (points indépendants), des sommets (points faisant partie d'une forme plus complexe, comme une ligne), ou des étiquettes spatiales (du texte spatialisé, à afficher sur une carte, par exemple, les noms des villes) ;
- les *arcs* (ou *polylignes*), définis par un nœud de début, un ensemble ordonné de sommets et un nœud de fin ;
- les *polygones*, une région fermée dont la frontière est définie par des *arcs* ;
- les *régions*, qui sont employées pour modéliser des zones complexes telles que des *polygones* qui peuvent se recouvrir, et des agrégats composés de plusieurs *polygones*.

Pour sa part, Oracle Spatial[2], un paquetage d'extensions qui peut être intégré au système de gestion de bases de donées Oracle 10g, supporte comme types de données spatiaux et graphiques :

- les *points bidimensionnels* ou *n-dimensionnels* qui peuvent être stockés, bien que l'indexation spatiale et les différents opérateurs spatiaux ne considèrent que les points à deux dimensions ;
- les *lignes*, composées de plusieurs points, liées par des lignes droites ou par des courbes ;
- les *polygones*, composés de plusieurs points, le premier et le dernier devant être identiques. Les polygones peuvent contenir des trous ;
- les *cercles*, définis par trois points colinéaires ;
- les *rectangles*, définis par deux points, le coin supérieur gauche et le coin inférieur droit.

Pour la manipulation de ces données, Oracle offre un ensemble assez complet d'opérateurs spatiaux.

Afin de combler ce problème d'hétérogénéité conceptuelle et implémentationnelle, les acteurs principaux de la communauté géomatique ont créé un consortium, appelé Open Geospatial Consortium (OGC) qui a pour but d'élaborer des standards permettant de rapprocher ces modèles, afin de permettre leur interopérabilité.

2.3.1.1 Le standard *OGC SFS* (*Simple Features Interface Standard*)

Les spécifications de l'OGC [1, 2, 129] sont destinées à être utilisées dans les SIG par les concepteurs, développeurs et utilisateurs d'informations géographiques, afin de manipuler et produire des jeux de données spatiaux. L'idée est de promouvoir l'interopérabilité dans le contexte du Web, des applications mobiles, des services localisés (Location Based Services ou LBS), ainsi que des solutions informatiques en général, pour des applications à référence spatiale.

La norme proposée par l'OGC en 1999 [1] fournit les schémas conceptuels pour la description et la manipulation des caractéristiques spatiales des différents éléments (en anglais, *features*) de l'espace géographique. Un élément (au sens conceptuel) représente une abstraction d'un phénomène du monde réel. Il est équivalent à la description thématique d'un objet géographique, évoquée dans la section 2.3.1. On parle d'élément géographique si ce dernier est associé à une localisation sur la Terre. Les données au format vectoriel regroupent les primitives géométriques et topologiques utilisées, séparément ou dans le même jeu de données, pour définir des objets exprimant les caractéristiques spatiales d'éléments géographiques. L'objectif est d'aboutir à une

[1] http ://www.esri.com/software/arcgis/arcinfo/about/features.html
[2] http ://www.oracle.com/technology/products/spatial/index.html

standardisation équivalente à celle existant pour le texte (ASCII), ou pour les nombres (IEEE), afin que les systèmes d'information traitent la géométrie d'une manière fiable et constante, comme ils le font pour le texte et les nombres.

Le modèle de classes pour la géométrie est illustré par le diagramme OMT de la figure 2.3. La classe abstraite racine de la hiérarchie (classe `Geometry`) a des sous-classes (également abstraites, à l'exception de la classe `Point`) pour représenter des points (classe `Point`), des courbes (classe `Curve`), des surfaces (classe `Surface`) et des collections de géométries quelconques (classe `GeometryCollection`). Chaque objet géométrique est associé à un système spatial de référence (classe `SpatialReferenceSystem`), qui décrit le système de coordonnées dans lequel l'objet géométrique est défini.

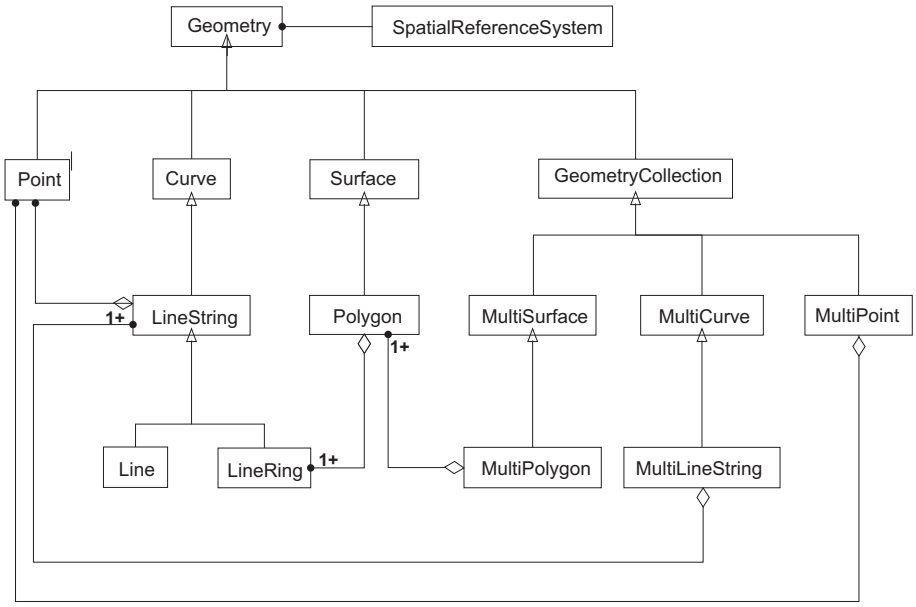


FIG. *2.3: Hiérarchie de types spatiaux dans le standard SFS de l'OGC.*

Ce diagramme étend le modèle géométrique fourni par les premières spécifications abstraites de l'OGC, par des classes de collections de géométries pour des objets à 0 (classe `MultiPoint`), 1 (classe `MultiLineString`), et 2 dimensions (classe `MultiPolygon`) pour modéliser des géométries correspondant aux collections respectivement de points, de polylignes (classe `LineStrings`) et de polygones (classe `Polygon`). `MultiCurve` et `MultiSurface` sont présentées comme des super-classes d'interface pour manipuler des collections de courbes et des surfaces. La figure 2.3 montre les agrégations entre les classes de collections de feuilles et leurs classes d'éléments.

Concrètement, la norme OGC [1] propose comme primitives pour la modélisation des *features* les types suivants :

a) *Point*, qui représente des géométries uniques dans l'espace à 0-dimension. Un point est spécifié à travers ses deux coordonnées x et y, et sa frontière est l'ensemble vide. Un exemple de point qui représente un endroit à Grenoble, dans le système de coordonnées WGS84 [3],[4], est (45.189084, 5.712504).

[3]Le WGS84 est le système géodésique associé au GPS, proposé en 2000 par NIMA [136]. Il s'est rapidement imposé comme une référence "standard" pour la cartographie.

[4]Tous les exemples de données spatiales présentés dans cette section, sont définis par rapport au système de coordonnées WGS84.

b) *MultiPoint*, qui est une collection dont les éléments sont de type *Point*, non ordonnés, ni connectés. Un exemple de *MultiPoint*, contenant deux points est ((45.189084, 5.712504), (45.189084, 75.712504)).

c) *Curve*, qui modélise des objets géométriques à une dimension, stockés en utilisant une séquences de points.

d) *LineString, Line, LinearRing* sont des spécialisations du type *Curve*. Le *LineString* représente les courbes pour lesquelles chaque paire de points consécutifs définit un segment de droite. Les droites, modélisées à l'aide du type *Line*, peuvent être vues comme des *LineString* ayant exactement deux points. Le type *LineRing* est équivalent au type *LineString* avec les contraintes de fermeture et simplicité satisfaites. De façon générale, une courbe est fermée si son point de départ est égal à son point d'arrivée, et elle est simple si elle ne passe pas deux fois par le même point. Un exemple de *LineString* qui contient trois points est *((45.189084, 5.712504), (45.189084, 75.712504), (58.569084, 35.540974))*.

e) *MultiCurve*, qui représente une collection de géométries dont les éléments sont de type *Curve*.

f) *MultiLineString*, une spécialisation de *MultiCurve* qui contient des éléments de type *LineString*.

g) *Surface*, qui modélise des objets géométriques dans l'espace à deux dimensions.

h) *Polygon*, une spécialisation du type *Surface*, défini par une frontière extérieure et 0 ou plusieurs frontières intérieures. Chaque frontière intérieure représente un trou dans le polygone. Les polygones sont fermés et leurs frontières sont spécifiées à l'aide d'objets de type *LineRings*. L'intérieur de chaque polygone est un ensemble connecté de points, mais l'extérieur d'un polygone avec 1 ou plusieurs trous n'est pas connecté. Un exemple de polygone est ((45.189084, 5.712504), (45.189084, 75.712504), (45.189084, 5.712504))).

i) *MultiSurface* est une collection de géométries à deux dimensions dont les éléments sont de type *Surface*.

j) *MultiPolygon* est une spécialisation de *MultiSurface*, dont les éléments sont de type *Polygon*. Un exemple de collection de polygones est ((45.189084, 5.712504), (43.189084, 75.712504), (45.189084, 5.712504)),((-5.569084,-45.540974), (-6.569084,-43.540974), (-5.569084,-45.540974))).

Afin d'accéder à des raisonnements qui exploitent les données modélisées à l'aide des types présentés ci-dessus, l'OGC propose un ensemble d'*opérateurs topologiques* : `Disjoint`, `Touches`, `Crosses`, `Within`, `Contains`, `Intersects` et `Overlaps` [1]. Ces derniers sont des fonctions formellement définies à partir des notions d'*intérieur* (I), de *frontière* (B) et d'*extérieur* (E) d'objets géométriques, permettent de déterminer la configuration spatiale existant entre deux objets spatiaux. Par exemple, l'opérateur *Disjoint* est défini comme suit :

$$a.Disjoint(b) \Longleftrightarrow a \cap b = \emptyset$$
$$a.Disjoint(b) \Longleftrightarrow (I(a) \cap I(b) = \emptyset) \wedge (I(a) \cap B(b) = \emptyset) \wedge (B(a) \cap I(b) = \emptyset) \wedge (B(a) \cap B(b) = \emptyset).$$

Ces types de représentation et de raisonnement sont à la base des outils de types SIG. Cependant, il est généralement reconnu qu'une représentation quantitative n'est pas très adaptée à la façon humaine de raisonner, qui est plus abstraite. Dans la section suivante, nous nous intéressons au domaine de la représentation spatiale qualitative, qui est plus proche de la façon humaine de percevoir l'espace.

2.3.2 Représentation spatiale qualitative

Si les représentations quantitatives permettent de réaliser des calculs efficaces et robustes impliquant les géométries attachées aux objets modélisés, elles se montrent plus adaptées lorsque les données géométriques sont incomplètes ou imprécises.

Représentation qualitative appliquée pour le domaine du spatial

Si la représentation purement qualitative a été très prisée pour la construction de nombreux systèmes physiques, son succès a été nettement limité dans le domaine du spatial. Selon [67, 65], l'espace étant, par définition, multidimensionnel, il ne peut pas être modélisé de façon adaptée en utilisant une seule quantité scalaire. Même si une telle représentation est possible, l'absence de mécanismes de raisonnement spatial purement qualitatifs fait que l'exploitation de ces modèles est extrêmement difficile. Forbus *et. al* [94] identifient la *transitivité* comme étant la caractéristique la plus importante des relations qualitatives, mais ils soulignent le fait qu'elle ne peut pas être bien exploitée dans des espaces à plusieurs dimensions.

De fait, il existe un grand nombre de travaux proposant des solutions pour la représentation qualitative des différents aspects de l'espace comme *la topologie, l'orientation, la distance, la forme, etc.* Deux visions différentes par rapport aux primitives sur lesquelles les théories spatiales doivent être construites ont été promues par les différents représentants de cette communauté de recherche.

La première vision est représentée par les théories mathématiques de l'espace, qui sont traditionnellement basées sur l'utilisation du *point* comme entité spatiale primitive, et sur la définition d'entités étendues (telles que les *régions*) comme des ensembles de points.

La méréologie ou le calcul des individus rejette le choix du *point* comme entité primitive et le classifie comme *erreur philosophique* [67], pour la raison que "*personne n'a jamais perçu un point, ni ne pourra jamais le percevoir, alors qu'on peut percevoir des individus ayant des extensions finies*" (traduction de Simons [220][5]). Ainsi, la deuxième tendance dans la communauté est de considérer la *région* comme élément primitif, et d'argumenter ce choix en soulignant le fait que l'extension spatiale de tout objet du monde réel est de type *région*. Les *points* peuvent être définis, si nécessaire, en termes de *régions*.

Une solution de compromis est proposée par le *calcul INCH* [106] qui traite les points et les régions comme des spécialisations d'une notion plus générale, celle d'*entité étendue*. Le but est d'étendre l'expressivité de l'algèbre RCC (décrite dans la section 2.3.2.1) tout en contournant les critiques sur le manque d'intuitivité des approches centrées sur les ensembles de points.

Une autre question ontologique vise la nature de l'espace de référence. Ce dernier pourrait être de façon conventionnelle $\mathbb{R}^n$, pour un n quelconque, mais l'on peut imaginer des applications pour lesquelles des univers discrets, finis ou non convexes seraient utiles.

Systèmes de référence

En général, une relation spatiale qualitative est définie par rapport à un système de référence, pour les ambiguïtés sur le sens de la relation [62]. Trois types de systèmes de référence sont distingués dans la littérature [211] :

[5] "No one has ever perceived a point, or ever will do so, whereas people have perceived individuals of finite extent."

- les systèmes de référence *intrinsèques* sont établis à partir d'un objet de référence qui détermine l'origine du système de coordonnées, aussi bien que son orientation.
- les systèmes de référence *extrinsèques* peuvent également hériter leur origine d'un objet de référence. Cependant, leur orientation est déterminée par des facteurs externes, tels que la direction du mouvement, ou par un objet conventionnel utilisé comme point de repère.
- les systèmes de référence *déictiques* impliquent trois objets : un objet primaire, un objet de référence, et un point de vue.

Raisonnement spatial qualitatif

Afin de réaliser des inférences à partir de relations spatiales qualitatives connues, on leur attache un "comportement" décrit à l'aide d'*algèbres de relations*. Ces algèbres formalisent certaines caractéristiques particulières des relations qui sont utilisées pour la déduction d'informations implicites. Par exemple, il est très utile de savoir qu'une relation spatiale r est *symétrique*, parce que si r est respectée par deux objets spatiaux A et B, alors on peut déduire qu'elle est également valable pour B et A. Dans la même optique, si on a deux relations spatiales r_1 et r_2 qui sont *inverses*, en sachant que les objets A et B respectent la relation r_1, on peut déduire que, entre B et A, la relation spatiale r_2 est valable. L'opération la plus puissante est, selon [217], la *composition*. Elle permet d'inférer une relation spatiale r_1 entre les objets A et C à partir de la connaissance des relations spatiales $A\, r_2\, B$ et $B\, r_3\, C$. La *transitivité* est un cas particulier de composition où les trois relations spatiales sont égales : $r_1 = r_2 = r_3$.

Dans cette thèse, nous avons étudié trois types de relations spatiales qualitatives : les relations *topologiques*, de *direction* et de *distance*, que nous présentons par la suite. Pour chaque type de relation spatiale, nous illustrons également les raisonnements qui peuvent être utilisés pour l'inférence des relations implicites.

2.3.2.1 Relations topologiques

Les approches les plus connues pour la modélisation des relations spatiales topologiques considèrent les *régions* plutôt que les *points* comme structures de base, et gagnent ainsi l'appellation de "*pointless geometries*". On peut citer notamment les travaux de Clarke [60] qui ont conduit au développement des algèbres RCC (*Region Connection Calculus*) [207, 66], elles-mêmes ultérieurement reprises et améliorées. La théorie de Clarke définit comme notion de base la connexion entre deux régions de l'espace, x et y, et décrit cette connexion en utilisant la relation symétrique et réflexive $C(x,y)$. Dans les algèbres RCC, l'interprétation de la relation $C(x,y)$ est légèrement changée pour exprimer le fait que la *fermeture* de deux régions partage un point [207]. La primitive $C(x,y)$ est très puissante, car elle peut être utilisée pour définir formellement tout un ensemble de fonctions et de prédicats spatiaux qui capturent des relations topologiques intéressantes et utiles, caractérisant différents degrés de connexion entre deux régions. La palette de relations définie par [207] inclut les situations suivantes :

- les régions sont *déconnectées* : $DC(x,y)$
- les frontières des régions sont *connectées* : $EC(x,y)$
- les régions sont *partiellement superposées* : $PO(x,y)$
- la région x est *partie tangentielle propre* de la région y : $TPP(x,y)$
- la région x est *partie tangentielle nonpropre* de la région y : $NTPP(x,y)$

– la région x est *partie propre* de la région y : $PP(x,y)$
– la région x fait *partie* de la région y : $P(x,y)$
– la région x est *égale* à la région y : $EQ(x,y)$
– les régions x et y se *superposent* : $O(x,y)$
– les régions x et y sont *connectées* : $C(x,y)$
– la région x est *discrète* par rapport à la région y : $DR(x,y)$.

Les huit relations de base – DC, EC, PO, TPP, $NTPP$, EQ, TPP_i, $NTPP_i$ – illustrées par la figure 2.4, représente un ensemble particulier de relations topologiques connu sous le nom d'*algèbre RCC-8*[6]. Une autre algèbre très connue est RCC-5, qui renonce à la distinction entre la frontière et l'intérieur d'une *région*, et considère en conséquence égales, les relations DC et EC, ainsi que les relations TPP et NTPP.

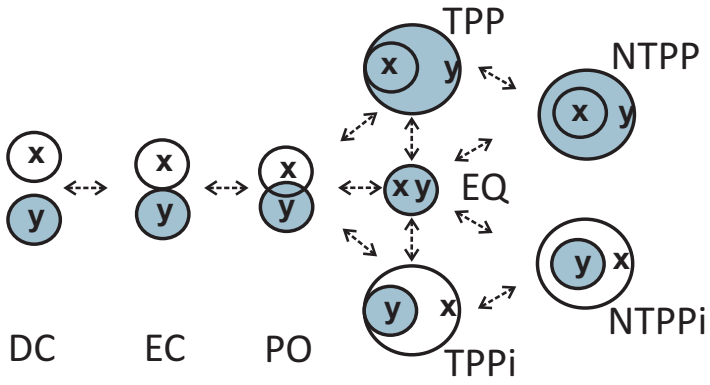


FIG. *2.4: Illustration des relations de l'algèbre RCC-8 et de leurs transitions continues.*

2.3.2.2 Raisonnement à base de relations topologiques

À partir des définitions de base et d'une connaissance partielle sur la position des régions, il est souvent nécessaire de pouvoir déduire d'autres connaissances sur la position des régions ou de vérifier la cohérence des connaissances établies [147]. Cela peut se réaliser grâce aux propriétés inter-relations, parmi lesquelles on peut citer :

– les relations *inverses* : on appelle relation inverse de la relation R la relation R_i telle que : $\forall (x,y) \in \left(\mathbb{R}^2\right)^2, R(x,y) \leftrightarrow R_i(y,x)$; Les relations TPP et $NTPP$ du calcul RCC8 ont comme relations inverses la relation TPP_i et la relation $NTPP_i$, respectivement.
– les relations *disjointes* : on dit que les relations R_1 et R_2 sont disjointes si $\forall (x,y) \in \left(\mathbb{R}^2\right)^2$, $R_1(x,y) \leftrightarrow \neg R_2(x,y)$.
– les relations *complémentaires* : on appelle relation complémentaire de R la relation R^C telle que : R et R^C sont disjointes et $\forall (x,y) \in \left(\mathbb{R}^2\right)^2$, $R(x,y) \vee R^C(x,y)$ est vrai. Les huit relations du calcul RCC8 sont disjointes et complémentaires deux à deux.
– la *composée* de deux relations R_1 et R_2 est la disjonction R de toutes les relations possibles entre x et z, étant données les trois régions x, y, z et $R_1(x,y)$ et $R_2(y,z)$ qui respectent les contraintes :

$$\forall (x,y,z) \in \left(\mathbb{R}^2\right)^3, R_1(x,y) \wedge R_2(y,z) \rightarrow R(x,z)$$
$$\text{et } \forall R_3, (R_1(x,y) \wedge R_2(y,z) \wedge R_3(x,z)) \rightarrow (R_3(x,z) \rightarrow R(x,z))$$

[6] En anglais : RCC-8 Calculus.

La dernière propriété est particulièrement intéressante, car, en utilisant des règles de composition (telles que celles illustrées dans le tableau de composition de la figure 2.6), on peut déduire toutes les relations possibles entre deux régions x et z à partir de la connaissance des relations entre ces régions prises séparément, avec une troisième région y [147]. Par exemple, les régions x, y, et z, illustrées dans la figure 2.5 sont telles que $TPP(x,y)$ et $EC(y,z)$. On peut donc déduire, que les régions x et z peuvent être reliés par une des deux relations suivantes : $DC(x,z)$ ou $EC(x,z)$. Ces deux configurations possibles sont illustrées dans la figure 2.5.

FIG. *2.5: Les deux situations possibles décrites par $TPP(x,y)$ et $EC(y,z)$.*

Les règles de composition des relations spatiales sont représentées à l'aide de tableaux de composition. La figure 2.6 illustre le tableau de composition pour les relations topologiques de l'algèbre RCC-8. Souvent, les relations inférées en utilisant un tableau de composition sont imprécises et représentées comme des disjonctions de plusieurs relations possibles. C'est le cas de l'exemple précèdent (voir figure 2.5), où le résultat de la composition des relations $TPP(x,y)$ et $EC(y,z)$ est $DC(x,z) \vee EC(x,z)$. Un autre exemple est la composition des relations EC et DC, qui peut donner DC, EC, PO, TPP_i ou $NTPP_i$: $EC \circ CD = DC \vee EC \vee PO \vee TPP_i \vee NTPP_i$ (voir la figure 2.6)

O	DC	EC	PO	TPP	NTPP	TPPi	NTPPi	EQ
DC	*	DC, EC, PO, TPP, NTPP	DC, EC, PO, TPP, NTPP	DC, EC, PO, TPP, NTPP	DC, EC, PO, TPP, NTPP	DC	DC	DC
EC	DC, EC, PO, TPPi, NTPPi	DC, EC, PO, TPP, TPPi, EQ	DC, EC, PO, TPP, NTPP	EC, PO, TPP, NTPP	PO, TPP, NTPP	DC, EC	DC	EC
PO	DC, EC, PO, TPPi, NTPPi	DC, EC, PO, TPPi, NTPPi	*	PO, TPP, NTPP	PO, TPP, NTPP	DC, EC, PO, TPPi, NTPPi	DC, EC, PO, TPPi, NTPPi	PO
TPP	DC	DC,EC	DC, EC, PO, TPP, NTPP	TPP, NTPP	NTPP	DC, EC, PO, TPP, TPPi, EQ	DC, EC, PO, TPPi, NTPPi	TPP
NTPP	DC	DC	DC, EC, PO, TPP, NTPP	NTPP	NTPP	DC, EC, PO, TPP, NTPP	*	NTPP
TPPi	DC, EC, PO, TPPi, NTPPi	EC, PO, TPPi, NTPPi	PO, TPPi, NTPPi	PO, TPP, TPPi, EQ	PO, TPP, NTPP	TPPi, NTPPi	NTPPi	TPPi
NTPPi	DC, EC, PO, TPPi, NTPPi	PO, TPPi, NTPPi	PO, TPPi, NTPPi	PO, TPPi, NTPPi	PO, TPP, NTPP, TPPi, NTPPi, EQ	NTPPi	NTPPi	NTPPi
EQ	DC	EC	PO	TPP	NTPP	TPPi	NTPPi	EQ

FIG. *2.6: Le tableau de composition pour les relations de l'algèbre RCC-8 : en ligne est donnée la relation entre les régions x et y ; en colonne celle entre les régions y et z ; en résultat les relations possibles entre x et z.*

Randell *et al.* [207] définissent une collection d'axiomes (voir le tableau 2.1) qui caractérisent les relations topologiques et sur laquelle est construit le treillis relationnel de subsomption illustré

dans la figure 2.7.

Relation topologique		Définition axiomatique
$DC(x,y)$	$\equiv_{def}$	$\neg C(x,y)$
$P(x,y)$	$\equiv_{def}$	$\forall z\,[C(z,x) \rightarrow C(z,y)]$
$PP(x,y)$	$\equiv_{def}$	$P(x,y) \wedge \neg P(y,x)$
$O(x,y)$	$\equiv_{def}$	$\exists z\,[P(z,x) \wedge P(z,y)]$
$DR(x,y)$	$\equiv_{def}$	$\neg O(x,y)$
$EC(x,y)$	$\equiv_{def}$	$C(x,y) \wedge \neg O(x,y)$
$TPP(x,y)$	$\equiv_{def}$	$P(x,y) \wedge \exists z\,[EC(z,x) \wedge EC(z,y)]$
$NTPP(x,y)$	$\equiv_{def}$	$P(x,y) \wedge \nexists z\,[EC(z,x) \wedge EC(z,y)]$
$EQ(x,y)$	$\equiv_{def}$	$P(x,y) \wedge P(y,x)$
$PO(x,y)$	$\equiv_{def}$	$O(x,y) \wedge \neg P(x,y) \wedge \neg P(y,x)$
$P^{-1}(x,y)$	$\equiv_{def}$	$P(y,x)$
$PP^{-1}(x,y)$	$\equiv_{def}$	$PP(y,x)$
$TPP^{-1}(x,y)$	$\equiv_{def}$	$TPP(y,x)$
$NTTP^{-1}(x,y)$	$\equiv_{def}$	$NTTP(y,x)$

TAB. *2.1: Définition axiomatique des relations topologiques (source [207]).*

Dans ce qui suit, nous rappelons la définition du treillis topologique, $\mathcal{T}_{\mathcal{P}}$, donnée par [149].

Définition 2.1. *Le treillis $\mathcal{T}_{\mathcal{P}}$ est la structure $\langle \mathcal{P}(\mathcal{B}), \subseteq, \bigcap, \bigcup, \bot, \top \rangle$ où :*
- *l'ensemble des éléments est l'ensemble des parties de $\mathcal{B}$ ($\mathcal{B}$ dénote l'ensemble de relations du RCC-8), noté $\mathcal{P}(\mathcal{B})$;*
- *l'ordre sur les éléments est l'inclusion ensembliste $\subseteq$;*
- *l'infimum de deux éléments R_1 et R_2 est obtenu par l'intersection des ensembles R_1 et R_2 ;*
- *le supremum de deux éléments R_1 et R_2 est obtenu par l'union $\bigcup$ des ensembles de R_1 et R_2 ;*
- *le minorant universel est l'ensemble $\bot$;*
- *le majorant universel est l'ensemble $\top$.*

Ce treillis permet de réaliser des inférences sur la composition des relations. La composition de toutes les relations se déduit à partir de la table de composition sur l'ensemble des relations (dénoté par $\mathcal{B}$) et du treillis de la figure 2.7 (dénoté par $\mathcal{T}_{\mathcal{P}}$) [149]. Par exemple, si on veut calculer la relation topologique qui existe entre deux objets spatiaux x et z, en partant des relations que les deux respectent par rapport à un troisième objet y : $PP(x,y)$ et $EC(y,z)$, on peut déduire que x et z sont liées par la relation $DR(x,z)$, comme suit :

$$PP(x,y) = TPP(x,y) \vee NTTP(x,y) \text{ (treillis) (voir 2.7)}$$
$$TPP(x,y) \circ EC(y,z) = DC(x,z) \vee EC(x,z) \text{ (tableau de composition) (figure 2.6)}$$
$$NTPP(x,y) \circ EC(y,z) = DC(x,z) \text{ (tableau de composition) (figure 2.6)}$$
$$PP(x,y) \circ EC(y,z) = DC(x,z) \vee EC(x,z) = DR(x,z) \text{ (treillis) (voir 2.7)}$$

Toutes les approches RCC ont en commun le fait que les relations qu'elles modélisent sont axiomatisées et définies en utilisant la *logique du premier ordre* qui leur fournit une sémantique formelle. Cependant, les procédures de décision basées sur cette formalisation ne sont pas très efficaces (problème, en général, NP-complet) [210, 209]. À partir des travaux qui ont porté sur les

propriétés formelles de ces théories [116, 202], il est même possible de déduire que les raisonnements qu'elles offrent sont incomplets et ne sont pas décidables.

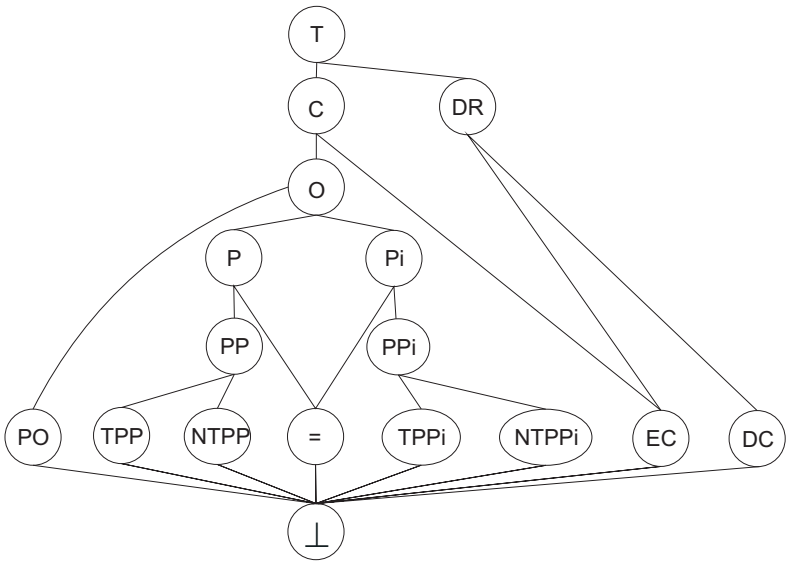


FIG. *2.7: Un treillis des relations topologiques de RCC-8 [149].*

2.3.2.3 Calcul des relations topologiques à partir de données quantitatives

Pour un nombre important d'applications se pose le problème du calcul des relations topologiques, que ce soit dans des basses de données spatiales ou sur des images [147]. Une approche en ce sens a été proposée par Egenhofer [87, 90, 85]. Elle consiste à attacher à chaque *région x*, trois ensemble de *points* : l'*intérieur* (x°), la *frontière* (∂x) et le *complément* (x^{-1}) (voir figure 2.8).

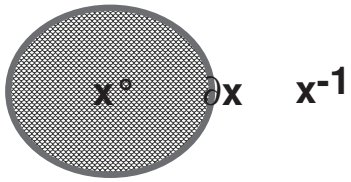


FIG. *2.8: La décomposition d'une région x en trois ensemble de points : l'intérieur (x°), la frontière (∂x) et le complément (x^{-1}).*

En utilisant ces trois ensembles, pour n'importe quelle paire de régions de l'espace, x et y, on peut construire une matrice 3x3 qui caractérise la relation topologique existant entre x et y :

$$I_n(x,y) = \begin{pmatrix} \partial x \bigcap \partial y & \partial x \bigcap y^\circ & \partial x \bigcap y^{-1} \\ x^\circ \bigcap \partial y & x^\circ \bigcap y^\circ & x^\circ \bigcap y^{-1} \\ x^{-1} \bigcap \partial y & x^{-1} \bigcap y^\circ & x^{-1} \bigcap y^{-1} \end{pmatrix}$$

Les éléments des matrices $I_i, i \in [1..512]$ font partie de l'ensemble $\{\emptyset, \neg\emptyset\}$. La valeur $\emptyset$ est utilisée lorsque l'intersection des ensembles étudiés (*e.g.* ∂x et y°, pour la ligne 1 et la colonne 2) est vide, et $\neg\emptyset$ est utilisée dans le cas contraire. Concrètement, si l'ensemble de points qui représente la frontière de l'objet x est disjoint de l'ensemble de points de la frontière de y ($\partial x \bigcap \partial y = \emptyset$), alors

l'élément $I_n[1,1] = \emptyset$, *etc.* Par exemple, la situation illustrée dans la figure 2.9 peut être modélisée à l'aide de la matrice I_7, du tableau 2.2.

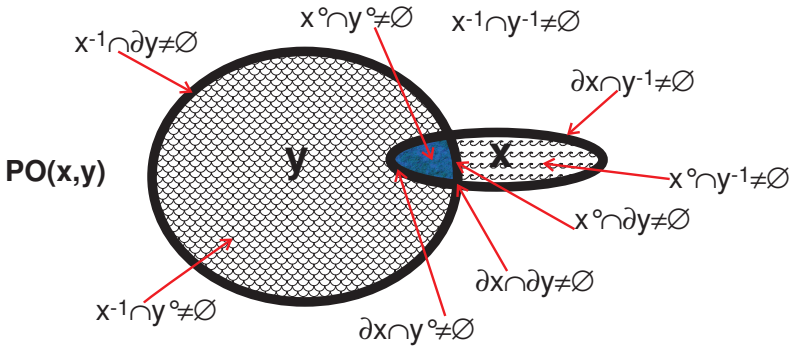


FIG. *2.9: Le calcul de la relation topologique entre les régions x et y, à l'aide de la méthode des 9 intersections.*

L'ensemble initial de $2^9 = 512$ matrices possibles est ramené à 8 si on prend en compte la réalité physique de l'espace 2D et des hypothèses spécifiques sur la nature des régions telles que : *i)* $x,y \subseteq \mathbb{R}^n$; *ii)* $x,y \neq \emptyset$; *iii)* tous les éléments de $\partial x, x^\circ, x^{-1}$ respectivement $\partial y, y^\circ, y^{-1}$ sont connectés ; *iv)* $x = \overline{x^\circ}$ et $y = \overline{y^\circ}$, ou $\bar{x}$ dénote la fermeture de la région x, obtenue comme intersection de tous les sous-ensembles fermés de l'ensemble x. Ces restrictions excluent les objets contenant des trous et les objets déconnectés. Les huit matrices obtenues, illustrées dans le tableau 2.2, correspondent de près aux relations de l'algèbre RCC-8.

$$I_0(x,y) = \begin{pmatrix} \emptyset & \emptyset & \neg\emptyset \\ \emptyset & \emptyset & \neg\emptyset \\ \neg\emptyset & \neg\emptyset & \neg\emptyset \end{pmatrix} \qquad I_1(x,y) = \begin{pmatrix} \neg\emptyset & \emptyset & \neg\emptyset \\ \emptyset & \emptyset & \neg\emptyset \\ \neg\emptyset & \neg\emptyset & \neg\emptyset \end{pmatrix} \qquad I_2(x,y) = \begin{pmatrix} \neg\emptyset & \emptyset & \emptyset \\ \emptyset & \neg\emptyset & \emptyset \\ \emptyset & \emptyset & \neg\emptyset \end{pmatrix}$$

$$a) R_{disjoint}(x,y) \equiv DC(x,y) \qquad b) R_{meet}(x,y) \equiv EC(x,y) \qquad c) R_{equal}(x,y) \equiv EQ(x,y)$$

$$I_3(x,y) = \begin{pmatrix} \emptyset & \neg\emptyset & \emptyset \\ \emptyset & \neg\emptyset & \emptyset \\ \neg\emptyset & \neg\emptyset & \neg\emptyset \end{pmatrix} \qquad I_4(x,y) = \begin{pmatrix} \neg\emptyset & \neg\emptyset & \emptyset \\ \emptyset & \neg\emptyset & \emptyset \\ \neg\emptyset & \neg\emptyset & \neg\emptyset \end{pmatrix} \qquad I_5(x,y) = \begin{pmatrix} \emptyset & \emptyset & \neg\emptyset \\ \neg\emptyset & \neg\emptyset & \neg\emptyset \\ \emptyset & \emptyset & \neg\emptyset \end{pmatrix}$$

$$d) R_{inside}(x,y) \equiv NTPP_i(x,y) \quad e) R_{coveredBy}(x,y) \equiv TPP(x,y) \quad f) R_{contains}(x,y) \equiv NTPP(x,y)$$

$$I_6(x,y) = \begin{pmatrix} \neg\emptyset & \emptyset & \neg\emptyset \\ \neg\emptyset & \neg\emptyset & \neg\emptyset \\ \emptyset & \emptyset & \neg\emptyset \end{pmatrix} \qquad I_7(x,y) = \begin{pmatrix} \neg\emptyset & \neg\emptyset & \neg\emptyset \\ \neg\emptyset & \neg\emptyset & \neg\emptyset \\ \neg\emptyset & \neg\emptyset & \neg\emptyset \end{pmatrix}$$

$$g) R_{covers}(x,y) \equiv TPP_i(x,y) \qquad h) R_{overlap}(x,y) \equiv PO(x,y)$$

TAB. *2.2: Les spécifications des huit relations topologiques en utilisant la méthode des 9 intersections, d'après Egenhofer, et leur équivalence en terme de relations du calcul RCC8.*

Les propriétés de ces relations sont étudiées dans [87] qui fournit également leur tableau de composition. Contrairement aux algèbres RCC, l'approche d'Egenhofer ne propose pas de sémantique formelle pour les relations qu'elle définie. Ainsi, l'étude des propriétés de complétude, de traçabilité et/ou de décidabilité de la théorie est difficile [209]. Toutefois, le travail de Grigni *et al.* [111] relève le défi en étudiant la complexité des raisonnements topologiques selon trois niveaux de précision différents, décrits comme des *niveaux de résolution*. Chacun de ces trois niveaux

de résolution est défini par rapport à un sous-ensemble de l'ensemble de relations topologiques proposé par Egenhofer, qui varie du niveau le plus simple contenant exclusivement les relations *overlap* et *disjoint*, jusqu'au niveau le plus complexe qui contient les huit relations. Les auteurs considèrent trois types d'expressions topologiques : *explicites* (la relation entre deux régions *A* et *B* est connue), *conjonctives* (la relation entre *A* et *B* est exprimée comme conjonction de deux ou plusieurs relations possibles) et *générales* (on ne dispose pas d'information sur la relation entre *A* et *B*, qui peut être chacune des huit relations possibles). Les auteurs vérifient également la satisfiabilité des *clauses* construites à partir de ces expressions. La vérification est faite selon deux axes : *i)* la *consistance relationnelle* - équivalente à un problème de satisfaction de contraintes (usuellement, des problèmes NP complets) imposées par les tableaux de composition des relations - et *ii)* l'existence d'une *réalisation* - une clause topologique est réalisable s'il existe un modèle concret de l'espace 2D qui est consistant du point de vue relationnel avec cette clause [111]. En conclusion, les auteurs montrent que le problème de la décidabilité des raisonnements topologiques est NP-complet dans les cas généraux, et décrivent quelques situations particulières pour lesquelles les algorithmes d'inférence sont polynomiaux.

2.3.2.4 Relations de direction

Afin de caractériser la position d'un objet spatial par rapport à un autre, on utilise des catégories qualitatives, telles que : "à droite du bâtiment", " au nord de Paris", "au sud de la France", "sur la table", *etc.*, en fonction des dimensions de l'univers du discours. Si le positionnement se fait par rapport à une échelle micro (qui peut aller de quelques centimètres jusqu'à 1 kilomètre), les catégories qualitatives employées le plus fréquemment sont : "en face", "derrière", "à gauche", et "à droite", rapportées soit à la position de l'observateur, soit à un objet de référence. Lorsqu'on se positionne par rapport à des échelles plus larges (meso – entre 1 km et 100 km, macro – entre 100 km et 10000 km et mega – plus de 10000 km) pour modéliser de manière équivalente la direction (ou l'orientation), on utilise les directions cardinales (*Nord*, *Sud*, *Est* et *Ouest*). Une correspondance peut être définie entre les deux ensembles, car ils traduisent les mêmes relations mais dans des contextes différents.

L'*orientation* est donc vue comme une relation ternaire, impliquant l'*objet caractérisé*, l'*objet de référence* et le *système de référence*. Selon [107], il existe trois types de systèmes de référence :

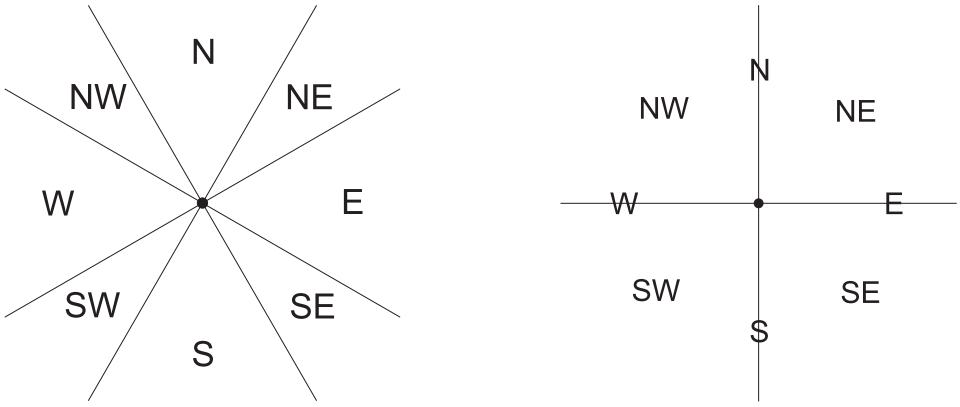
a) un système de référence intrinsèque organisé selon les parties *avant*, *arrière*, *droite*, *gauche* intrinsèques aux objets modélisés ;

b) un système relatif à la position de l'observateur ;

c) un système de référence extrinsèque, défini par la position des pôles sur la Terre.

La plupart des approches qui traitent de l'*orientation* considèrent comme entités primitives les *points* et limitent les calculs à l'espace à deux dimensions. Dans cette catégorie, on trouve le travail de Frank [96] qui propose, pour définir l'orientation, le partitionnement de l'espace en neuf régions. L'idée est de considérer des objets représentés par des points et de partitionner l'espace en plusieurs zones qui correspondent aux différentes classes de direction (*Nord, Sud, Ouest, NordEst, etc.*) et dont l'origine est l'*objet de référence*. Frank décrit deux types de partitionnement de l'espace : l'un qui définit des régions de type cône (voir figure 2.10a), et l'autre basé sur la notion de projection (voir figure 2.10b). Pour un partitionnement du premier type, la relation *Nord(B)* fait référence à une région en forme de cône, dont le sommet est l'objet *B*, ayant un angle de 45°

et un axe dont la direction est le *Nord*. La deuxième approche interprète la même relation comme étant exactement la demi-droite dont l'origine est l'objet *B* et qui représente le Nord géographique. En revanche, les relations intermédiaires (comme *NordEst*, *SudOuest*, *etc.*) sont vues comme des régions et non pas des droites. Pour les deux approches, Frank introduit une relation neutre *O*, qui signifie que les objets sont trop proches l'un de l'autre pour qu'on puisse établir la relation de direction qui les lie.



(a) Partitionnement de l'espace dans des cônes.

(b) Partitionnement de l'espace à base de projection.

FIG. *2.10: Relations d'orientation entre des objets représentés par des points.*

Une approche différente, appelée la *double intersection*, a été proposée par Freksa [97] afin de modéliser des relations d'orientation ternaires, comportant le *point à localiser* (*c*), le *point de référence* (*b*), et le *point de perspective* (*a*). Elle consiste en la division de l'espace 2D en 8 régions. D'abord, en utilisant comme orientation de référence la droite *ab* on divise l'espace en deux semi-plans perpendiculaires à *ab* : la partie *droite* et la partie *gauche*. Ensuite, une segmentation *devant*/*derrière* est réalisée, argumentée par le constat que la plupart des entités du monde réel telles que les maisons, les gens, les animaux, *etc.*, ont une partie *devant*/*derrière* intrinsèque. Finalement l'auteur identifie l'ensemble de huit relations disjointes illustré dans la figure 2.11 : *devant*, *devant-droite*, *droite-neutre*, *derrière-droite*, *derrière*, *derrière-gauche*, *gauche-neutre* et *devant-gauche*. Freksa propose également un tableau de composition pour les relations qu'il définit.

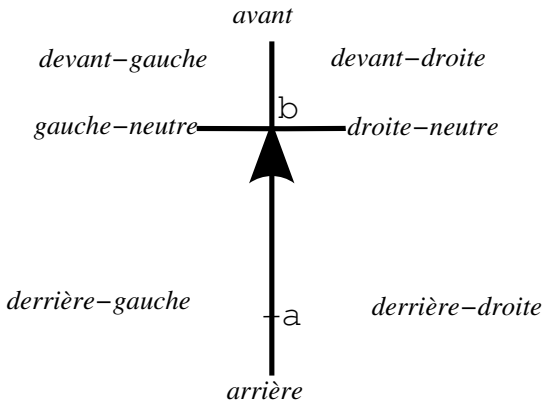


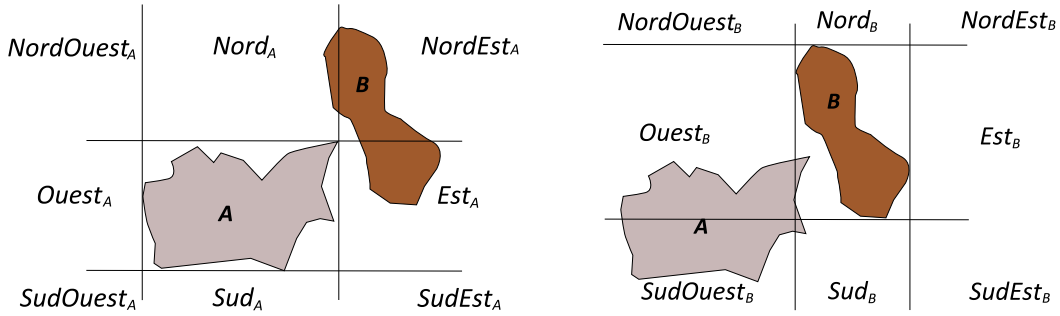
FIG. *2.11: Le système d'orientation proposé par Freksa.*

Le problème devient encore plus complexe si l'on considère des objets étendus représentés par des polygones [86, 107], non seulement parce qu'ils peuvent croiser plusieurs secteurs lors d'une comparaison avec d'autres objets étendus, mais aussi parce qu'ils disposent de directions intrinsèques (par exemple, on parle du nord d'un département). Le plus souvent, pour déterminer la relation de direction entre deux régions représentées par des polygones, on utilise des primitives d'abstraction comme, par exemple, le *rectangle englobant minimal* ou le centre géométrique.

Dans sa thèse, Goyal [107] adapte le modèle propose par Egenhofer pour la représentation des relations topologiques pour les relations de distance entre des objets étendus. Son approche, appelée *PDR*[7], est basée sur la méthode de la projection, appliqué au rectangle englobant minimal de l'*objet de référence* pour partitionner l'espace en neuf *carreaux*. Les *carreaux* périphériques correspondent aux huit directions : *Nord*, *Nord-Est*, *Sud*, etc. tandis que le carreau du milieu coïncide avec la direction *neutre*. Dans ce système, la relation de direction *basique* entre l'*objet de référence* et un *objet cible* est donnée par une matrice carrée qui enregistre les carreaux qui sont croisés par la géométrie de l'*objet cible*.

$$dir_{RR}(A,B) = \begin{pmatrix} NW_A \bigcap B & N_A \bigcap B & NE_A \bigcap B \\ W_A \bigcap B & O_A \bigcap B & E_A \bigcap B \\ SW_A \bigcap B & S_A \bigcap B & SE_A \bigcap B \end{pmatrix}$$

Cette méthode est beaucoup plus détaillée que les méthodes qui emploient les abstractions des objets. Cependant, son inconvénient réside dans la perte de la convertibilité. Considérons, par exemple, deux objets, A et B, ayant une relation B $(Nord_A, NordEst_A, Est_A)$ par rapport à A. En utilisant une approximation des objets par des rectangles, on déduit que la relation de direction entre A et B est $(Sud_B, SudOuest_B, Ouest_B)$. Néanmoins, dans le cas où on utilise les géométries exactes des objets, en fonction de la forme de A les relations suivantes peuvent être déduites : (Sud_B, $Neutre_B$, $Ouest_B$), ($SudOuest_B$, $Neutre_B$, $Ouest_B$), (Sud_B, $Neutre_B$, $SudOuest_B$) ou bien (Sud_B, $Neutre_B$, $SudOuest_B$, $Ouest_B$) (figure 2.12b).



(a) Les relations de direction entre A et B, en prenant A comme *objet de référence*.

(b) Les relations de direction entre A et B, en prenant B comme *objet de référence*.

FIG. *2.12: Relations de direction entre objets étendus, exprimées en utilisant la méthode proposée par Goyal.*

[7] *Projection-based Directional Relations*

$$dir_{RR}(A,B) = \begin{pmatrix} \emptyset & \neg\emptyset & \neg\emptyset \\ \emptyset & \emptyset & \neg\emptyset \\ \emptyset & \emptyset & \emptyset \end{pmatrix} \quad dir_{RR}(B,A) = \begin{pmatrix} \emptyset & \emptyset & \emptyset \\ \neg\emptyset & \neg\emptyset & \emptyset \\ \neg\emptyset & \neg\emptyset & \emptyset \end{pmatrix}$$

TAB. *2.3: Les matrices de direction pour les objets illustrés dans la figure 2.12*

Goyal propose également une matrice de direction détaillée qui garde pour chaque *carreau* de direction le pourcentage de la surface de l'*objet cible* qui le recouvre. Ce raffinement implique des valeurs comprises entre 0% pour une intersection vide et 100% si et seulement si l'objet est entièrement inclus dans un seul *carreau*, et dont la somme est toujours 1 (voir l'exemple de la figure 2.4).

$$dir(A,B) = \begin{pmatrix} 0 & 0.05 & 0.60 \\ 0 & 0 & 0.35 \\ 0 & 0 & 0 \end{pmatrix} \quad dir_{RR}(B,A) = \begin{pmatrix} 0 & 0 & 0 \\ 0.39 & 0.06 & 0 \\ 0.52 & 0.03 & 0 \end{pmatrix}$$

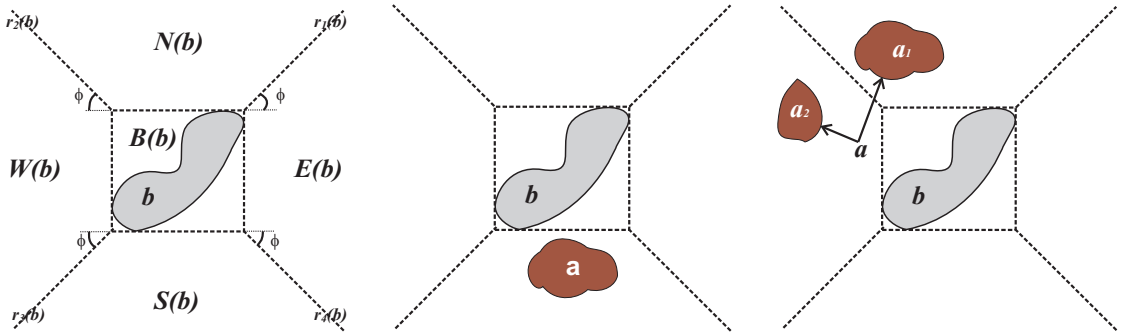
TAB. *2.4: Les matrices de direction étendues pour les objets illustrés dans la figure 2.12*

Une autre approche pour la modélisation des relations de direction binaires, appelée méthode *CDR*, a été récemment proposée par [224]. Elle peut être vue comme une combinaison entre la méthode *PDR* et la méthode des cônes proposée par Frank [96]. La théorie de Skiadopoulos *et al.* traite des objets spatiaux qui représentent des régions (modélisées à l'aide des polygones) de l'espace 2D. Une région est définie comme un ensemble non-vide et borné de points dans $\mathbb{R}^2$.

Notation 2.1. *Deux types de régions sont prises en compte par ce modèle :*

- *les régions REG, pour lesquelles un homomorphisme peut être établi avec le disque de dimension 1 et fermé ($\{(x,y) : x^2+y^2 \leq 1\}$). Les régions REG sont fermées, connectées, et ont des frontières connectées. ;*
- *les régions REG^*, construites comme union finie de régions REG, qui peuvent être déconnectées et peuvent avoir des trous.*

Comme la méthode de projection PDR [107], la méthode *CDR* utilise le rectangle minimal englobant pour approximer l'objet de référence. L'espace autour de l'objet de référence est divisée en 5 régions appelées *carreaux* en utilisant la méthode des cônes (voir figure 2.13a). Les carreaux périphériques correspondent aux quatre relations de direction : *nord*, *sud*, *est* et *ouest*, désignées par $N(b)$, $S(b)$, $E(b)$ et $W(b)$ respectivement. Le carreau central correspond au rectangle minimal englobant ($mbb(b)$) de b, et est désigné par $B(b)$. En fonction des dimensions de l'angle ϕ utilisé pour la construction des carreaux, une infinité de modèles peuvent être définis. Chaque modèle de la famille *CDR* sera identifié par une valeur unique de l'angle ϕ ($0^\circ \leq \phi \leq 90^\circ$), appelé *angle caractéristique.*



(a) La division de l'espace en 5 carreaux, à partir de l'approximation de l'objet *b* par son rectangle minimal englobant.

(b) Relation de type "carreau unique" entre l'objet *a* et l'objet de référence *b*.

(c) Relation de type "carreau multiple" entre l'objet déconnecté *a* et l'objet *b*.

FIG. *2.13: Relations de direction entre l'objet de référence b et l'objet caractérisé a, modélisées à l'aide de la méthode CDR*

Pour les régions de type REG^* (voir notation 2.1), les auteurs introduisent la notion de *relation de direction basique*, à l'aide de la définition 2.2. Une relation de direction basique est de type *carreau unique* si $k = 1$ (voir figure 2.13b) et de type *carreaux multiples* dans le cas inverse (voir figure 2.13c). Pour éviter la confusion entre différentes relations de type *carreaux multiples*, les carreaux faisant partie d'une relation basique sont cités dans l'ordre préétabli : N, W, S, E, B. Par exemple, la relation $N : W : B$ est une relation valide alors que $W : N : B$ ou $N : B : W$ ne le sont pas.

Définition 2.2. *Une relation de direction basique est une expression* $R_1 : \ldots : R_k$*, où :*

a) $1 \leq k \leq 5$,

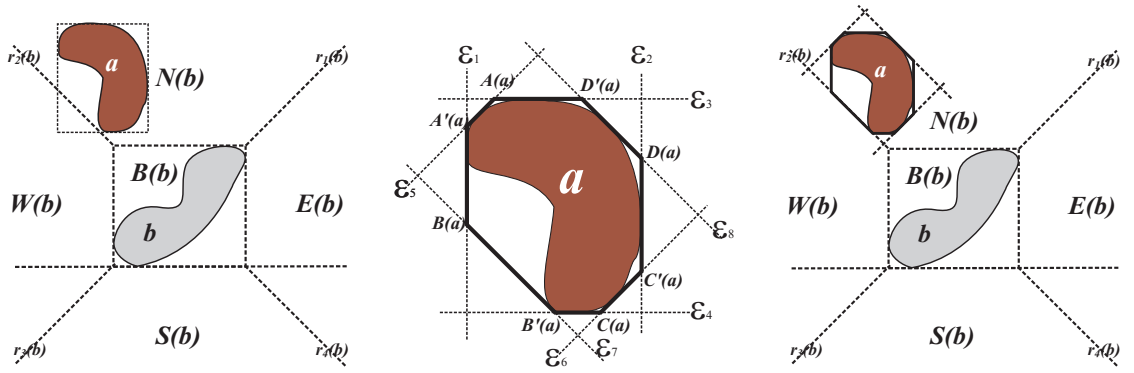
b) $R_1, \ldots, R_k \in \{N, W, S, E, B\}$ *et*

c) $R_i \neq R_j, \forall i,\ j$ *tel que* $1 \leq i, j \leq k$ *et* $i \neq j$.

Les modèles *CDR* proposent un ensemble exhaustif de 31 relations basiques (noté B^*), qui sont disjointes deux à deux. Le nombre de relations manipulées par le modèle *CDR* est beaucoup moins important en comparaison des 511 relations obtenues avec le modèle *PDR* [224]. Les *relations basiques* sont utilisées pour représenter les connaissances explicites et précises sur la direction entre deux régions. Les relations imprécises, sont décrites par rapport à l'ensemble 2^{B^*}. Par exemple, la relation imprécise "l'objet *a* est soit en partie au *nord* et en partie à l'*ouest* par rapport à l'objet *b*, soit entièrement à l'ouest de *b*", est décrite par $a\ \{N : W, N\}\ b$. En général, l'expression $a\ \cup_{i=1}^{n} R_i\ b$ indique le fait que l'objet *a* est relié à l'objet *b* par une relation parmi l'ensemble $R_1, \ldots, R_n$.

Pour définir formellement une *relation de direction basique* de type *carreau unique*, Skiadopoulos *et al.* utilisent l'abstraction de l'*objet cible* par l'octogone minimal englobant (voir figure 2.14b), car, dans certains cas, le rectangle englobant minimal utilisé par *PDR* introduit du bruit, comme illustré dans la figure 2.14a, où la relation déduite entre *a* et *b* est $a\ N : W\ b$ alors qu'elle devrait être $a\ N\ b$. L'octogone englobant minimal (voir figure 2.14b) est une polygone défini par les lignes $\varepsilon_1, \ldots, \varepsilon_8$, où :

– $\varepsilon_1, \ldots, \varepsilon_8$ sont tangentes à l'objet *a* ;



(a) La relation de direction $N : W$ obtenue en utilisant le rectangle englobant minimal de a.

(b) La construction de l'octogone englobant minimal.

(c) La relation de direction N obtenue en utilisant l'octogone englobant minimal de a.

FIG. *2.14: Relations de direction entre l'objet de référence b et l'objet caractérisé a, calculées en utilisant l'abstraction de a par un rectangle englobant minimal et par l'octogone englobant minimal.*

- $\varepsilon_1, \varepsilon_2$ sont parallèles à l'axe Oy et $(\varepsilon_1)_x < (\varepsilon_2)_x$[8],
- $\varepsilon_3, \varepsilon_4$ sont parallèles à l'axe Ox et $(\varepsilon_3)_y < (\varepsilon_4)_y$,
- $\varepsilon_5, \varepsilon_6$ forment un angle ϕ avec l'axe Ox et $(\varepsilon_5)_x < (\varepsilon_6)_x$,
- $\varepsilon_7, \varepsilon_8$ forment un angle de $180^\circ - \phi$ avec l'axe Ox et $(\varepsilon_7)_x < (\varepsilon_8)_x$.

Définition 2.3. *Dans un modèle CDR(ϕ) arbitraire($0^\circ < \phi < 90^\circ$), les relations N, W, S, E et B entre deux objets a et b $\in REG^*$ sont définies comme suit :*

$$
\begin{array}{lll}
a\ N\ b & ssi & \tan(\phi) B_x(a) + B_y(a) \geq \tan(\phi) B_x(b) + A_y(b), \\
 & & \tan(\phi) C_x(a) - C_y(a) \leq \tan(\phi) D_x(b) - A_y(b)\ et \\
 & & C_y(a) \geq A_y(b) \\
a\ W\ b & ssi & \tan(\phi) D_x(a) + D_y(a) \leq \tan(\phi) B_x(b) + A_y(b), \\
 & & \tan(\phi) C_x(a) - C_y(a) \leq \tan(\phi) B_x(b) - C_y(b)\ et \\
 & & D_x(a) \leq B_y(b) \\
a\ S\ b & ssi & \tan(\phi) A_x(a) - A_y(a) \geq \tan(\phi) B_x(b) - C_y(b), \\
 & & \tan(\phi) D_x(a) + D_y(a) \leq \tan(\phi) D_x(b) + C_y(b)\ et \\
 & & A_y(a) \leq C_y(b) \\
a\ E\ b & ssi & \tan(\phi) A_x(a) - A_y(a) \geq \tan(\phi) D_x(b) - A_y(b), \\
 & & \tan(\phi) B_x(a) + B_y(a) \geq \tan(\phi) D_x(b) + C_y(b)\ et \\
 & & B_x(a) \geq D_x(b) \\
a\ B\ b & ssi & C_y(a) \geq C_y(b), \\
 & & A_y(a) \leq A_y(b), \\
 & & B_x(a) \geq B_x(b)\ et \\
 & & D_x(a) \leq D_x(b).
\end{array}
$$

où $A(a) = \varepsilon_4|\varepsilon_5$[9], $A'(a) = \varepsilon_1|\varepsilon_5$, $B(a) = \varepsilon_1|\varepsilon_7$, $B'(a) = \varepsilon_3|\varepsilon_7$, $C(a) = \varepsilon_3|\varepsilon_6$, $C'(a) = \varepsilon_2|\varepsilon_6$, $D(a) = \varepsilon_2|\varepsilon_8$ *et* $D'(a) = \varepsilon_4|\varepsilon_8$.

[8] Par $(\varepsilon)_x$ (respectivement $(\varepsilon)_y$) on dénote la coordonnée x (respectivement y) du point d'intersection de la ligne ε avec l'axe Ox (respectivement Oy).

[9] Par $\varepsilon_1|\varepsilon_2$, on dénote le point d'intersection entre les lignes ε_1 et ε_2.

En partant des 5 relations de type *carreau unique* Skiadopoulos *et al.* introduisent les relations de direction de type *carreaux multiples* à l'aide de définition 2.4.

Définition 2.4. *Pour deux objets spatiaux a et b* $\in REG^*$*, la relation* $R = R_1 : \ldots : R_k$ *décrit la position de a par rapport à b si et seulement si* $\exists a_1, \ldots, a_k \in REG^*$ *tel que* $a_1\ R_1\ b, a_2\ R_2\ b, \ldots, a_k\ R_k\ b$ *et* $a = a_1 \cup \ldots \cup a_k$.

2.3.2.5 Raisonnement à base de relations de direction

Afin de pouvoir exploiter les relations de direction dans un raisonnement qualitatif, Frank [96] fournit des tableaux de composition (voir figure 2.15) pour chacun des deux systèmes de direction qu'il propose. Ces tableaux soulignent le fait que l'ajout de la relation d'identité *O* augmente considérablement l'efficacité des déductions. Si, en l'absence de l'identité seulement 8 des 64 compositions peuvent être résolues, avec l'identité on obtient des conclusions, bien que pas toujours exactes, pour chacune des 81 paires de relations. La comparaison des deux tableaux révèle la méthode à base de projection comme étant plus précise, car le nombre de raisonnements approximatifs qu'elle produit (32) est inférieur à celui produit par la méthode à base de cônes (56) [96].

	N	NE	E	SE	S	SW	W	NW	O
N	N	n	ne	o	o	o	nw	nw	N
NE	n	ne	ne	e	e	o	n	n	NE
E	ne	ne	E	e	se	o	o	o	E
SE	o	e	e	SE	se	s	o	o	SE
S	o	o	se	se	S	s	sw	w	S
SW	o	o	o	s	s	SW	sw	w	SW
W	nw	o	o	o	sw	sw	W	w	W
NW	n	n	o	o	o	w	w	NW	NW
O	N	NE	E	SE	S	SW	W	NW	O

(a) Tableau de composition pour les directions à base de cône.

	N	NE	E	SE	S	SW	W	NW	O
N	N	N	NE	e	o	w	NW	NW	N
NE	NE	NE	NE	e	e	o	n	n	NE
E	NE	NE	E	SE	SE	s	o	n	E
SE	e	e	SE	SE	SE	s	s	o	SE
S	o	e	SE	SE	S	SW	SW	w	S
SW	w	o	s	s	SW	SW	SW	w	SW
W	NW	n	o	s	SW	SW	W	NW	W
NW	NW	n	n	o	w	w	NW	NW	NW
O	N	NE	E	SE	S	SW	W	NW	O

(b) Tableau de composition pour les directions à base de projection.

FIG. *2.15: Comparaison des tableaux de composition pour les deux types de relations de direction proposées par Frank. Les relations dénotées par des minuscules sont des résultats approximatifs.*

L'approche basée sur la projection proposée par Frank [96] permet de représenter les neuf relations de direction en termes d'algèbre de points, en spécifiant, pour chacune des deux dimensions d'un point, la relation algébrique satisfaite par rapport au point de référence (l'origine du système de projection). Cela fournit à l'approche une sémantique formelle qui a été utilisée par Ligozat [155] pour définir les relations de direction en termes de paires de relations d'ordre (<,=,>) entre points, comme illustré dans le tableau 2.5. Pour l'algèbre de relations de direction qu'il obtient, Ligozat conclut que le problème de la vérification de la cohérence d'un réseau de contraintes est NP-complet. Cependant, il identifie une sous-classe de relations - les relations *pré-convexes* - pour lesquelles le même problème est décidable et d'une complexité $O(n^3)$.

Comme support pour le raisonnement à base de relations de direction étendues, définies selon la méthode des neufs intersections proposée par Goyal et Egenhofer [108, 107, 109], une méthode de composition a été proposée par les auteurs dans [109]. Cependant, Skiadopoulos et Koubarakis montrent dans [223] que la méthode de composition étudiée par Goyal et Egenhofer [109] n'est pas

O	*Nord*	*Est*	*Sud*	*Ouest*	*Nord-Est*	*Nord-Ouest*	*Sud-Est*	*Sud-Ouest*
$(=,=)$	$(=,>)$	$(>,=)$	$(=,<)$	$(<,=)$	$(>,>)$	$(<,>)$	$(>,<)$	$(<,<)$

TAB. *2.5: La définition algébrique proposée par Ligozat pour les relations de direction.*

correcte dans tous les cas. Suite à ce constat, ils proposent dans [223] une nouvelle formalisation pour les relations de direction étendue ainsi que plusieurs tableaux de composition.

Pour la méthode *CDR*, Skiadopoulos *et al.* définissent dans [224] des algorithmes pour déduire l'inverse d'une *relation de direction élémentaire*, ainsi que pour composer deux relations de direction. Ils partent du constat de la symétrie inhérente au modèle *CDR* et du lemme 2.1 qui établit l'inverse de chaque relation de type *carreau unique* en vérifiant toutes les configurations spatiales possibles.

Lemme 2.1. *Si $R \in \{N,W,S,E\}$ est une relation de direction de type carreau unique, alors*

a) $inv(R) = \delta\left(R^{\leftarrow}, R^{\rightarrow}, R^{\downarrow}\right) - \{R^{\leftarrow}, R^{\rightarrow}\}$ *et*

b) $inv(B) = U_{dir} - \{N,W,S,E\}$[10].

Notation 2.2. *La fonction $\delta(R_1,\ldots,R_k)$ désigne l'ensemble de relations de direction basiques qui peuvent être construites en combinant les relations de type carreau unique $R_1,\ldots,R_k$.*

Par exemple,

$$\delta\{N,W,B\} = \{N,W,B,N:W,N:B,W:B,N:W:B\}.$$

Les expressions $R^{\leftarrow}$, $R^{\rightarrow}$ et $R^{\downarrow}$ désignent le carreau obtenu en se déplaçant à partir du carreau R, $R \in \{N,W,S,E\}$, dans le sens contraire aux aiguilles d'une montre (respectivement, dans le sens des aiguilles d'une montre, et dans le sens diamétral opposé). Ces expressions sont définies par le tableau 2.6.

R	N	W	S	E
$R^{\leftarrow}$	W	S	E	N
$R^{\rightarrow}$	E	N	W	S
$R^{\downarrow}$	S	E	N	W

TAB. *2.6: La définition des expressions R, $R \in \{N,W,S,E\}$ (source [224]).*

Pour obtenir l'inverse d'une relation de direction de type *carreaux multiples*, Skiadopoulos *et al.* définissent le lemme 2.2. La preuve de celui-ci est également faite par des vérifications sur les configurations spatiales possibles. Finalement, pour trouver l'inverse d'une relation de direction arbitraire, on utilise le théorème 2.3.

Lemme 2.2. *L'inverse d'une relation de direction de type carreaux multiples $R = R_1 : \ldots : R_k$ $(2 \leq k \leq 5)$ est calculée en utilisant l'une des deux fonctions suivantes :*

a) $inv(R) = \delta\left(\overline{R}\right)$ *si* $B \notin \{R_1,\ldots,R_k\}$ *et*

b) $inv(R) = \delta\left(\overline{R},B\right) - \overline{R}$ *si* $B \in \{R_1,\ldots,R_k\}$,

[10] U_{dir} représente la relation de direction universelle : $U_{dir} = \delta(N,W,S,E,B)$.

où $\overline{R} = \{N,W,S,E,B\} - \{R_1,\ldots,R_k\}$

Théorème 2.3. *Pour une relation de direction* $Q = \cup_{i=1}^{k}(R_i)$, $Q \in 2^{B^*}$, *l'inverse est définie comme étant* $inv(Q) = \cup_{i=1}^{k}(inv(R_i))$. *L'ensemble* $inv(R_i)$ *est calculé à l'aide des lemmes 2.1 et 2.2.*

Les auteurs proposent également des règles et des algorithmes pour la composition des relations de distance que nous n'illustrons pas ici, mais qui peuvent être consultés dans [224].

2.3.2.6 Relations de distance

La façon dont on perçoit l'espace, et donc les distances entre les objets, est fortement dépendante de notre expérience, ainsi que du contexte dans lequel on se situe. Par exemple, la sémantique de la relation ***proche de*** entre A et B ne dépend pas seulement des positions absolues de A et de B, mais aussi de leurs dimensions et de leur forme, de la position d'autres objets, du système de référence, du temps qu'on met pour aller de A à B, *etc.*

De plus, il existe deux catégories de relations qualitatives qui visent la distance entre objets : les *distances absolues* et celles *relatives. La distance relative* est obtenue à partir d'une comparaison entre la distance considérée et une distance de référence et aboutit à une relation ternaire telle que *plus proche que, équidistant* ou *plus loin que*.

Hernandez *et al.* proposent de modéliser des relations de *distance absolues* dans un *framework* construit à partir de trois éléments : *l'objet primaire* (PO), *l'objet de référence* (RO) et le *système de référence* (FofR) [128]. Plusieurs types de distances peuvent être pris en compte : la distance à vol d'oiseau, le plus court chemin dans un réseau routier, ou encore le temps d'accès en fonction d'un moyen de transport déterminé (train, voiture, etc.). La façon la plus simple et la plus répandue de calculer la distance entre deux objets spatiaux est d'utiliser la distance euclidienne, définie dans l'espace 2D, à l'aide de la formule : $d(A,B) = \sqrt{(x_A - x_B)^2 + (y_A - y_B)^2}$. Pour les distances euclidiennes, on peut citer comme propriétés, la symétrie, la réflexivité et l'inégalité triangulaire :

$$\begin{gathered} d(A,B) = d(B,A)\,; \\ d(A,A) = 0\,; \\ d(A,B) + d(B,C) \geq d(A,C). \end{gathered}$$

Pour d'autres types de distance, ces propriétés ne s'appliquent pas forcément. Par exemple, dans un réseau routier, les autoroutes sont souvent plus rapides que les petites routes qui sont en revanche plus courtes. De plus, le même trajet entre A et B peut prendre des intervalles de temps différents en fonction de la direction du trajet : de A vers B ou l'inverse. C'est notamment le cas des routes de montagne, où la montée est généralement plus lente que la descente.

Les relations de distance peuvent être définies à différents niveaux de granularité. Les types d'objets comparés, ainsi que le contexte dans lequel les objets sont étudiés, sont des facteurs décisifs pour établir l'ensemble des distinctions pertinentes, et, donc, l'ensemble des relations qualitatives à utiliser. En général, le premier niveau de granularité qui vient à l'esprit sépare l'espace en deux régions circulaires : une région `proche` et une région `loin`, centrées sur l'objet de référence (voir figure 2.16). Mais le besoin de systèmes qui gèrent des relations d'une granularité plus fine est également mis en avant par [128].

Pour obtenir une distance qualitative à partir des coordonnées physiques des objets, on utilise une échelle qui divise la ligne des réels en plusieurs secteurs (par exemple : `très proche`, `proche`,

`médium`, `loin` et `très loin`) en fonction de la granularité souhaitée (voir figure 2.16). La valeur numérique obtenue lors du calcul numérique de la distance est alors positionnée sur cette échelle (*i.e.* dans un secteur) pour donner une distance qualitative absolue.

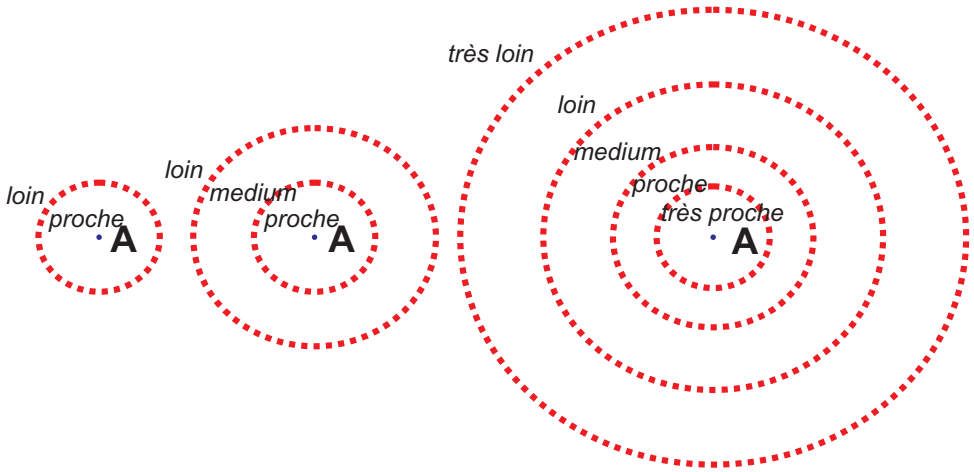


FIG. *2.16: Relations de distance définies selon plusieurs granularités.*

Clementini *et al.* [61] formalisent les relations de distance dans le cas général, où l'espace autour de l'objet de référence, RO, est partitionné en $n+1$ régions (la granularité est $n+1$). À chacune de ces régions on peut attacher un symbole de distance (*i.e.* `très proche`, `proche`, `médium`, *etc.*), donc on obtient $n+1$ symboles : $Q = \{q_0, q_1, \ldots, q_n\}$, où q_0 est la distance la plus proche de *RO* et q_n est la distance la plus éloignée de RO. Entre les symboles de distance, il existe un ordre total : $q_0 < q_1 < \ldots < q_n$. Étant donné un ensemble d'objets O, la distance qualitative entre *PO* et *RO*, $(PO, RO \in O)$ est une fonction $d : O \times O \rightarrow Q$, qui associe à *PO* le symbole de distance le plus petit qui identifie la distance qualitative de *PO* par rapport à *RO*. De façon générale, la distance qualitative entre un objet A et un objet B est exprimée par $d_{AB} = d(A, B)$.

Une autre distinction est faite dans [128] entre l'étendue de la plage de distance de rang i, dénotée par δ_i, et Δ_i qui dénote la plage de distance entre l'objet d'origine et la limite inférieure de la plage de distance de rang $i+1$ (voir figure 2.17). Il est important de noter le fait que le symbole q_i désigne toutes les distances entre l'origine et les points qui se situent dans la plage Δ_i.

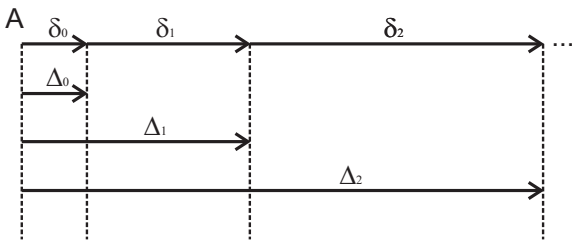


FIG. *2.17: Division de l'espace en plusieurs plages de distances à l'aide de facteurs de croissance.*

Une restriction très souvent imposée aux plages de distance est, selon [128], la croissance monotone :

$$\delta_0 \leq \delta_1 \leq \ldots \leq \delta_n. \tag{2.1}$$

Une autre restriction utile impose que la grandeur de chaque plage de distance de rang i soit supérieure à la somme des plages antérieures :

$$\delta_i \geq \Delta_{i-1}, \forall i > 0. \tag{2.2}$$

De plus, si une plage de distance δ_i est beaucoup plus grande qu'une plage δ_j qui la précède ($\delta_i \gg \delta_j$), alors un processus d'absorption est réalisé lors de la composition des deux distances :

$$\delta_i \pm \delta_j \simeq \delta_i \tag{2.3}$$

2.3.2.7 Raisonnement à base de relations de distance

Pour la déduction de distances implicites, Hernandez *et al.* ont défini des règles de compositions [128]. Ces règles permettent de déduire, à partir de la distance qualitative $d_{AB} = d(A,B)$, entre les objets A et B, et la distance qualitative $d_{BC} = d(B,C)$ entre les objets B et C, la distance qualitative possible entre A et C. En général, la distance qui peut résulter d'une opération de composition est décrite par un ensemble de distances possibles, pour lequel les auteurs définissent une limite inférieure, LB et une limite supérieure UB.

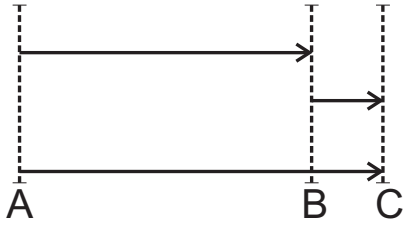


FIG. *2.18: Composition des distances entre des objets ayant la même direction* .

Dans le cas illustré dans la figure 2.18, où la direction de B par rapport à A est la même que la direction de C par rapport à B, composer les deux relations de distance se résume à l'addition de deux quantités positives. Dans ce cas, la limite inférieure (LB) de la plage dans laquelle se situe la distance résultante ne peut pas être inférieure au maximum des deux distances :

$$LB(d_{AC}) = d_{AB} \oplus d_{BC} = max(d_{AB}, d_{BC}).$$

Si aucune restriction structurelle n'est connue sur le système de distances traité, la limite supérieure de la distance d_{AC} est q_n. Si l'on traite des distances monotones croissantes (voir restriction 2.1), la limite supérieure de la plage dans laquelle s'inscrit la relation résultat d_{AC} est calculée par l'équation :

$$UB(d_{AC}) = ord^{-1}(ord(d_{AB}) + ord(d_{BC})).$$

La fonction ord est définie par : $ord : Q \rightarrow \{1,\ldots,n,n+1\}$, tel que $ord(q_i) = i+1$. Pour des indices i qui dépassent $n+1$ on a $ord^{-1}(i) = q_n$.

En considérant un espace divisé en trois secteurs de distance, le tableau 2.7 synthétise les résultats de la composition des relations de distance [128] dans le cas où les objets A et B et B et C ont la même orientation.

$\oplus$	q_0	q_1	q_2
q_0	q_0, q_1	q_1, q_2	q_2
q_1	q_1, q_2	q_1, q_2	q_2
q_2	q_2	q_2	q_2

TAB. *2.7: Tableau de composition des distances monotones croissantes pour la même direction.*

Si on considère la restriction 2.2, la limite UB de la plage dans laquelle se situe d_{AC} peut être restreinte à la plage qui suit le maximum des deux plages qui correspond aux distances composées :

$$UB(d_{AC}) = succ(max(d_{AB}, d_{BC})).$$

De plus, si la division de l'espace respecte la règle d'absorption 2.3 on peut restreindre davantage la limite supérieure de la plage dans laquelle la distance d_{AC} est incluse, l'effet de la distance la plus petite dans la composition étant annulé :

$$|ord(d_{AB}) - ord(d_{BC}))| \geq p \Rightarrow UB(d_{AC}) = max(d_{AB}, d_{BC}).$$

La composition des distances ayant des orientations opposés (voir figure 2.19), résulte en une plage de distance dont la limite supérieure (UB) est le maximum des deux distances considérées :

$$UB(d_{AC}) = d_{AB} \ominus d_{BC} = max(d_{AB}, d_{BC}).$$

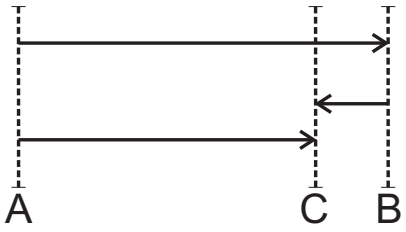


FIG. *2.19: Composition de distances entre des objets ayant des directions inverses.*

Si on n'impose pas de restrictions supplémentaires, la seule propriété qu'on peut déduire sur la limite inférieure de la plage de d_{AC} est son équivalence à q_0. Si on impose la restriction 2.1, LB devient :

$$LB(d_{AC}) = ord^{-1}(|ord(d_{AB}) - ord(d_{BC})|).$$

Le résultat de la composition de trois distances pour une orientation contraire est identique pour les restrictions 2.1 et 2.2. Celui-ci est illustré par le tableau 2.8.

$\ominus$	q_0	q_1	q_2
q_0	q_0	q_0, q_1	q_1, q_2
q_1	q_0, q_1	q_0, q_1	q_0, q_1, q_2
q_2	q_1, q_2	q_0, q_1, q_2	q_0, q_1, q_2

TAB. *2.8: Tableau de composition des distances monotones croissantes pour des directions inverses.*

2.4 Représentation du temps

Après ce passage en revue des techniques de représentation et de raisonnement reposant sur des données spatiales quantitatives et des relations spatiales qualitatives, à présent nous présentons les approches qui traitent de la représentation du temps et qui relèvent du *niveau informatique*. Concrètement, nous étudions la représentation temporelle quantitative à travers des types de données et opérations dédiés, et la représentation temporelle qualitative, mise en œuvre à travers des algèbres de relations qualitatives. Ces théories sont spécifiées formellement en ayant recours au *niveau mathématique* (voir section 2.2.1). Le *niveau utilisateur* sera abordé dans le chapitre 3.

2.4.1 Représentation temporelle quantitative

Contrairement à la situation de l'espace, en ce qui concerne le temps, les considérations se fixent rapidement sur une unité sans durée, l'*instant*, qui peut être assimilé à un point sur une droite, et sur une unité ayant une durée, l'*intervalle*, assimilable à un segment de droite [37]. Cette approche est basée sur une vision discrète du temps, qui permet d'identifier, par exemple, l'instant où une attaque terroriste a eu lieu, ou bien la période pendant laquelle Jacques Chirac a été Président de la République Française.

De même que pour la représentation spatiale, une norme internationale (ISO 8601) [236] a été définie par l'Organisation Internationale de Normalisation (ISO), pour la description des instants et des intervalles de temps, avec l'objectif d'éviter tout risque de confusion dans les communications internationales en raison du grand nombre de notations nationales différentes. La norme est construite sur quatre concepts fondamentaux :

a) la notion d'*instant* (en anglais *time-point*), qui définit un intervalle de temps indivisible. Un *instant* est spécifié par sa position (sa "distance" par rapport à l'instant zéro) sur la droite du temps.

b) la notion d'*intervalle* (en anglais *time-interval*), qui représente une durée de temps entre deux instants donnés, nommés *début* (*start*) et *fin* (*end*). Un intervalle peut être spécifié par les deux instants *début* et *fin* ou par l'un d'entre eux et la durée temporelle entre les deux.

c) la notion d'*intervalle répétitif*, qui représente une série d'intervalles de temps consécutifs ayant la même durée.

d) la notion de *durée*, qui représente une quantité de temps.

La norme traite le temps selon différents niveaux de détail en utilisant les structures suivantes :

- *Date* - la notation internationale pour une date est *YYYY-MM-DD*, où *YYYY* représente l'année définie par rapport au calendrier *grégorien*, *MM* représente le mois de l'année, ses valeurs étant entre 01 (janvier) et 12 (décembre) et DD représente le jour du mois (de '1' à '31').
- *Heure*, qui débute par la lettre *T* (pour Time, soit heure en anglais), suivie des éléments utiles, dans cet ordre : heures (de '00' à '24'), minutes (de '00' à '59') puis secondes ('00' à '59') en utilisant deux-points ' :' pour séparateur.
- *Heure locale avec décalage horaire* - les dates et le temps sont définis par rapport à un fuseau horaire. La norme offre la possibilité d'indiquer le fait qu'une heure est mesurée par rapport au temps universel (UTC), ou, dans le cas contraire, de spécifier le décalage horaire positif ou négatif. Ainsi, *T09 :00 :00+01 :00* est équivalent à *T08 :00 :00Z* (9 heures dans le fuseau ayant une heure de plus que l'UTC donc 8 heures en UTC).

– *Date et heure* - correspondent à l'heure spécifiée d'un jour spécifié.
– *Intervalle de temps* - il est également possible d'indiquer une durée (mesure d'un temps écoulé) ou un intervalle de temps (entre deux dates précises). Une durée représente une quantité de temps dans l'absolu, sans commencer à un instant précis. Par exemple, une durée de 18 ans, 9 mois, 4 jours, 11 heures, 9 minutes et 8 secondes est définie par l'expression :*P18Y9M4DT11H9M8S*.
  On peut également combiner les deux notations et indiquer une période en fonction de sa date de début et sa durée. Par exemple, *2004-12-02/P3D* est l'intervalle de temps d'une durée de 3 jours débutant le 2 décembre 2004.
– *Intervalle de temps périodique* - si la durée est répétitive, on précède la notation par la lettre R. Par exemple, R/PT01 signifie «toutes les heures».

L'ISO 8601 est utilisée dans les programmes informatiques, journaux de bord, inscriptions, rapports de magazine, pages Web, courrier électronique, statistiques, formulaires de toutes sortes, administrations et entreprises, douanes et transports, commerce électronique et universités, ainsi que dans tous les types d'activités internationales.

La norme est également à la base de la définition standardisé des types de données temporels à travers des Schéma XML [239, 38]. Les types temporels mis à disposition par le Schéma XML sont illustrés dans la figure 2.20. Le *Consortium World Wide Web* (W3C) a inclus certains de ces types dans ses recommandations pour les langages à ontologies pour le Web sémantique, RDF(S) et OWL. Une description détaillée de chacun de ces types est donnée dans [38].

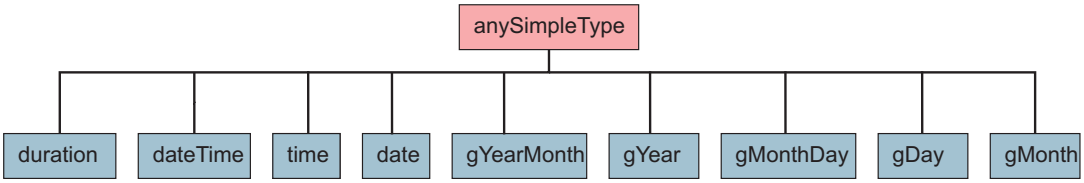


FIG. *2.20: Les types temporels définis par le Schéma XML (Source [38]).*

Le W3C définit, à travers la recommandation [38], un ensemble d'opérations pour les données de type temporel. Les durées (`duration`) par exemple, peuvent être comparées en utilisant les facettes de comparaison `minInclusive`, `minExclusive`, `maxInclusive` et `maxExclusive`. Cependant, les résultats de ces comparaisons ne sont pas toujours évidentes. C'est le cas des durées d'un mois (`P1M`) et de 30 jours (`P30D`). Ainsi, pour les durées, la recommandation établit un ordre partiel. C'est aussi le cas pour les types `time`, `gYear`, `gYearMonth`, `gMonthDay`, `gMonth`, `gDay`, pour lesquels il n'est pas toujours possible d'établir une relation d'ordre sans équivoque pour deux valeurs dont une possède une indication de fuseau horaire mais pas l'autre. Il est également possible de comparer les instants (`dateTime`) en utilisant leur conversion dans des valeurs décimales, réalisée par la fonction `timeOnTimeline`[38]. Entre les instants, la recommandation établit un ordre total.

L'opération d'addition entre un instant *S* et une durée *D* est aussi décrite dans [38]. Elle est très utile pour déterminer si un instant donné est inclus dans une période spécifique. Les durées (`duration`) peuvent être additionnées avec les valeurs de types `date`, `gYearMonth`, `gYear`, `gDay` et `gMonth`.

2.4.2 Représentation temporelle qualitative

La théorie de sens commun la plus populaire pour raisonner avec le temps est celle d'Allen [14]. Ce calcul d'intervalles a pour origine la nécessité de raisonner sur des processus ayant une certaine durée. Dans cette optique, Allen ne considère que des intervalles et ignore les instants [183]. Les relations proposées sont toutes dérivées d'une unique primitive, *meets*, qui exprime que deux intervalles se touchent sans se recouvrir. À partir de là, il est possible de définir un ensemble exhaustif de relations entre des intervalles convexes, illustré dans la figure 2.21 et dénoté par la suite par B_{int}. Cet ensemble de relations constitue selon [183] un "'standard"' en intelligence artificielle, et est utilisé dans de nombreux domaines tels que le traitement du temps en langage naturel, la planification, les raisonnements temporels divers, *etc.*

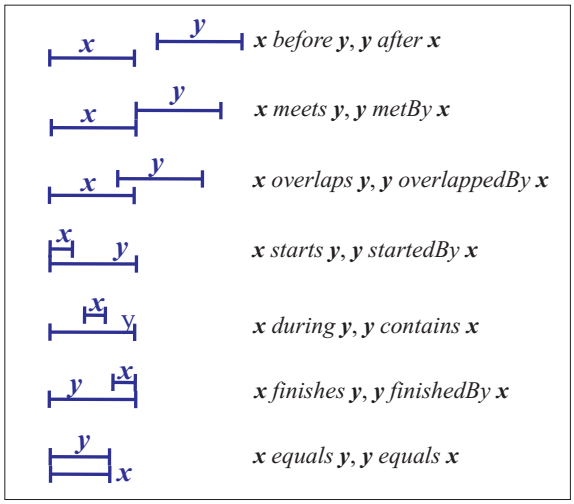


FIG. *2.21: Illustration des treize relations atomiques d'Allen.*

Tout couple d'intervalle satisfait au moins une des relations atomiques de B_{int}. De plus, il est facile de constater que deux intervalles satisfaisant une de ces relations ne peuvent pas satisfaire une autre. L'ensemble $2^{B_{int}}$ contient 2^{13}, soit 8192 d'éléments. Parmi ceux-ci, on peut citer deux relations particulières : la relation vide ($\{\}$), qui correspond à l'impossibilité de relier temporellement deux entités temporelles, et la relation totale (B_{int}), correspondant au fait qu'on ne possède aucune information temporelle relative à deux entités, dans ce cas-là toutes les relations atomiques sont envisageables.

2.4.3 Raisonnement à base de relations temporelles qualitatives

L'ensemble B_{int} est muni de trois opérations binaires : union ($\cup$), intersection ($\cap$) et composition ($\circ$) et de deux opérations unaires : inverse ($^{-1}$) et complémentaire ($^{\sim}$). L'union, l'intersection et le complémentaire ont la même sémantique que les opérations homonymes définies pour les ensembles [68].

Six paires de relations sont inverses (voir tableau 2.9) et la relation `equals` est sa propre inverse.

Relation			Complémentaire
before	(b)	(a)	after
meets	(m)	(M)	metBy
overlaps	(o)	(O)	overlappedBy
finishedBy	(F)	(f)	finishes
contains	(c)	(d)	during
starts	(s)	(S)	startedBy
equals	(e)	(e)	equals

TAB. *2.9: Les paires de correspondances relation-complémentaire pour les relations d'Allen.*

Le tableau de composition pour les relations atomiques fourni par Allen [14] est illustré dans le tableau 2.10.

○	b	m	o	F	c	s	e	S	d	f	O	M	a
b	(b)	(b)	(b)	(b)	(b)	(b)	(b)	(b)	(bmosd)	(bmosd)	(bmosd)	(bmosd)	full
m	(b)	(b)	(b)	(b)	(b)	(m)	(m)	(m)	(osd)	(osd)	(osd)	(Fef)	(cSOMa)
o	(b)	(b)	(bmo)	(bmo)	(bmoFc)	(o)	(o)	(oFc)	(osd)	(osd)	concur	(cSO)	(cSOMa)
F	(b)	(m)	(o)	(F)	(c)	(o)	(F)	(c)	(osd)	(Fef)	(cSO)	(cSO)	(cSOMa)
c	(bmoFc)	(oFc)	(oFc)	(c)	(c)	(oFc)	(c)	(c)	concur	(cSO)	(cSO)	(cSO)	(cSOMa)
s	(b)	(b)	(bmo)	(bmo)	(bmoFc)	(s)	(s)	(seS)	(d)	(d)	(dfO)	(M)	(a)
e	(b)	(m)	(o)	(F)	(c)	(s)	(e)	(S)	(d)	(f)	(O)	(M)	(a)
S	(bmoFc)	(oFc)	(oFc)	(c)	(c)	(seS)	(S)	(S)	(dfO)	(O)	(O)	(M)	(a)
d	(b)	(b)	(bmosd)	(bmosd)	full	(d)	(d)	(dfOMa)	(d)	(d)	(dfOMa)	(a)	(a)
f	(b)	(m)	(osd)	(Fef)	(cSOMa)	(d)	(f)	(OMa)	(d)	(f)	(OMa)	(a)	(a)
O	(bmoFc)	(oFc)	concur	(cSO)	(cSOMa)	(dfO)	(O)	(OMa)	(dfO)	(O)	(OMa)	(a)	(a)
M	(bmoFc)	(seS)	(dfO)	(M)	(a)	(dfO)	(M)	(a)	(dfO)	(M)	(a)	(a)	(a)
a	full	(dfOMa)	(dfOMa)	(a)	(a)	(dfOMa)	(a)	(a)	(dfOMa)	(a)	(a)	(a)	(a)

TAB. *2.10: La table de composition des relations atomiques d'Allen. Dans cette table, le terme full désigne les 13 relations (bmoFcseSdfOMa) et le terme concur désigne les 9 relations (oFcseSdfO), d'après [16].*

Trois ans après la publication de cette théorie, Vilain et Kautz [235] ont prouvé que le problème de la cohérence des réseaux d'intervalles est NP-complet dans le cas général. Suite à ce constat, de nombreuses études ont eu pour but l'identification et l'analyse des fragments de l'ensemble de relations pour lesquels le problème de la cohérence est polynomial [154, 187, 142].

2.5 Synthèse

La première partie de ce chapitre nous a permis de mettre en exergue les travaux que nous jugeons les plus importants dans le domaine de la représentation quantitative et qualitative de l'information spatiale. Nous avons étudié les différents ensembles de types spatiaux utilisés comme point de départ pour la construction des *Systèmes d'Information Spatiale*, ainsi que les opérateurs utilisés pour l'exploitation des informations spatiales. Nous avons également présenté le standard *OGC SFS*, dont le but est de définir un ensemble de types spatiaux *de référence* qui peut être utilisé par les concepteurs, les développeurs et les utilisateurs d'informations géographiques, afin de manipuler et produire des jeux de données spatiaux dans des scénarios d'interopérabilité.

Nous avons souligné le fait que la représentation quantitative, très adaptée lorsqu'il s'agit de réaliser des calculs efficaces et robustes qui impliquant les géométries attachées aux objets modélisés, se montre peu adaptée lorsque les données géométriques sont incomplètes ou imprécises. Pour résoudre ce problème, plusieurs approches de représentation qualitative des informations spatiales

ont été proposées par différentes communautés de chercheurs. Dans ce travail, nous avons étudié notamment les *relations spatiales topologiques*, de *direction* et de *distance*. Nous nous sommes intéressés aux différentes approches de représentation de ces relations spatiales qualitatives ainsi qu'à leurs propriétés axiomatiques qui peuvent être exploitées afin de déduire, à partir des relations spatiales explicites, les relations spatiales implicites.

Il faut notamment retenir l'algèbre RCC-8 [60, 207], dédiée à la représentation des relations topologiques entre des régions, l'approche *CDR* [224] pour la représentation des relations de direction entre des régions, ainsi que l'approche de Hernandez *et al.* [128] pour la modélisation des relations de distance. Nous allons utiliser ces relations ainsi que les méthodes de calcul et de raisonnement qui leur sont spécifiques pour la définition d'un moteur de raisonnement spatial, capable d'exploiter à la fois les informations spatiales quantitatives et les relations spatiales qualitatives afin de déduire des relations spatiales implicites (voir chapitre 6).

La deuxième partie de ce chapitre traite de la représentation quantitative et qualitative de l'information temporelle. Nous avons étudié la norme ISO 8601 [236], dédiée à la description des instants et des intervalles de temps et qui est à la base de la définition standardisée des types de données temporels à travers des Schéma XML. Après le passage en revue des types temporels proposés par le Schéma XML, nous avons présenté la théorie d'Allen, qui est l'approche qualitative la plus connues pour raisonner avec le temps.

Enfin, les relations temporelles d'Allen sont notre source principale d'inspiration pour la définition du raisonneur temporel proposé dans le chapitre 6.

Chapitre 3

Représentations pour le Web Sémantique Géospatial

I just had to take the hypertext idea and connect it to the TCP and DNS ideas and - ta-da ! - the World Wide Web.

Tim Berners-Lee

SOMMAIRE

Résumé

Ce chapitre présente les travaux que nous jugeons les plus importants pour la représentation spatiale et temporelle à l'aide d'ontologies OWL. Nous partons d'une étude générale sur le Web Sémantique, qui présente les langages proposés pour la représentation des ontologies, les outils de modélisation et les outils de raisonnement qui existent à ce jour. Ensuite, nous étudions ces langages et outils du point de vue de l'information spatiale et temporelle, et nous présentons les principaux défis pour la mise en place d'un Web Sémantique Géospatial.

Après une courte introduction sur les principes du Web Sémantique Géospatial, imaginé comme une extension spatio-temporelle du Web Sémantique, nous décrivons les travaux du groupe de travail Geospatial Incubator Group dont le but est de proposer des recommandations en vue de la standardisation d'ontologies géospatiales. Concernant la représentation du temps, nous présentons l'ontologie OWL-Time, recommandée par le W3C pour la description d'informations temporelles dans le cadre du Web Sémantique.

3.1 Introduction

Le Web Sémantique Géospatial [88, 93] regroupe les activités autour du développement d'ontologies spatio-temporelles pour la description sémantique d'informations à références spatiales et temporelles accessibles sur le Web. L'objectif du Web Sémantique Géospatial est similaire à celui du Web Sémantique : associer aux données spatio-temporelles des descriptions (méta-données) interprétables par les humains et surtout par les machines, afin que le traitement automatisé de ces données par des agents logiciels soit plus efficace. Comme pour le Web Sémantique, la plupart des travaux de recherche autour du Web Sémantique Géospatial portent sur l'intégration, l'interopérabilité et la recherche d'information à l'aide d'ontologies.

Selon Schell [213], le Web Géospatial et, en conséquence le Web Sémantique Géospatial, est un domaine complexe, situé à la croisée de plusieurs domaines de recherche (voir figure 3.1), parmi lequel on peut citer : la géographie physique, la géomatique, la représentation de connaissances, les réseaux, la logique, *etc.*

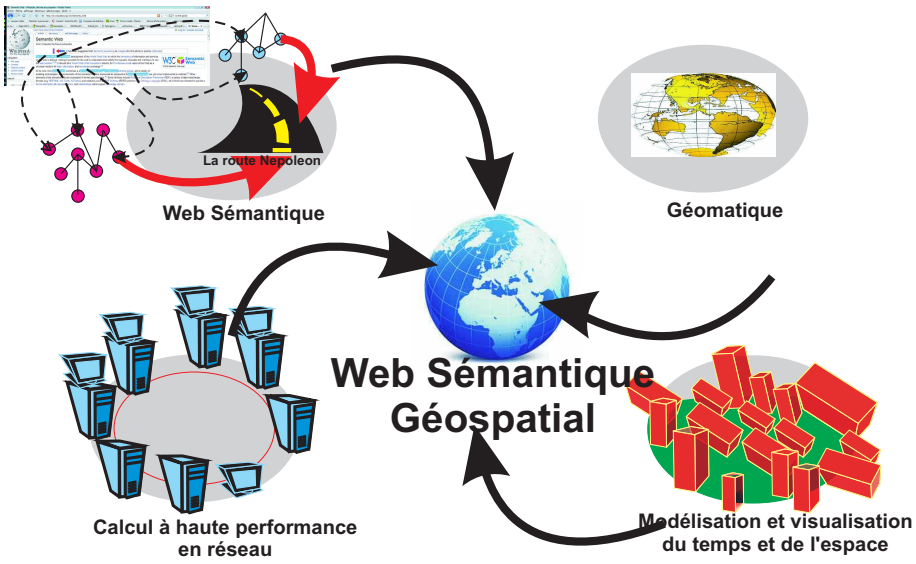


FIG. *3.1: Le Web Sémantique Géospatial à la croisée de multiples domaines de recherche.*

Concrètement, le Web Sémantique Géospatial peut être vu comme une extension du Web Sémantique qui garantit une gestion adaptée des annotations spatiales et temporelles quantitatives et qualitatives. Cette gestion est définie en termes de représentation et de raisonnement.

Dans la première partie de ce chapitre nous présentons les principes, les langages et les outils du Web Sémantique, sur lesquelles s'appuient également le Web Sémantique Géospatial. La deuxième partie du chapitre est concentré autour de la représentation des informations spatiales et temporelles (quantitatives et qualitatives) à l'aide d'ontologies.

3.2 Principes, langages et outils du Web Sémantique

Depuis le lancement du World Wide Web, au début des années 90, les sites Web ont connu un essor fulgurant, tant dans leur nombre (d'une trentaine de sites en 1992 à plus d'un trillion de sites fin 2008 [15]), que dans la diversité et la richesse de leur contenu. L'une des conséquences

de la forte croissance d'Internet et du Web est que leur exploitation est forcement dépendante des performances des moteurs de recherche. En effet, ces derniers sont devenus selon [72] des acteurs centraux dans l'infrastructure du Web. Une étude réalisée en décembre 2008 par Pew Internet & American Life Project [203] place l'utilisation des moteurs de recherche pour la découverte d'informations sur le Web, en deuxième position dans le classement des activités les plus fréquemment réalisées par les internautes américains. Les moteurs de recherche actuels retrouvent les documents sur le Web en fonction des correspondances syntaxiques qui existent entre leurs contenus textuels et des mots-clés donnés. Mais l'utilité de ces types d'approches de recherche est de plus en plus contestable, surtout si on prend en compte les phénomènes de synonymie (les documents qui contiennent des synonymes pour les mots clés spécifiés pour la requête ne sont pas inclus dans la liste des réponses) et de polysémie (certains mots clés peuvent avoir plusieurs sens, donc tous les documents qui contiennent ces mots clés ne sont pas forcément pertinents pour la requête donnée). On peut ajouter à cela l'important volume de données numériques disponibles sur le Web, qui fait que le nombre de réponses fourni pour une requête est souvent trop important pour que l'utilisateur puisse explorer ces résultats en totalité.

Le Web Sémantique [34] vise à offrir des solutions pour accroître la performance, évaluée en termes de *rappel* et de *précision*[1], des moteurs de recherche, en annotant le contenu des ressources Web à l'aide de concepts ontologiques compréhensibles et exploitables par les machines.

Le terme *Web Sémantique* fait d'abord référence à la vision du Web de demain comme un vaste espace d'échange de ressources entre êtres humains et agents logiciels permettant une exploitation, qualitativement supérieure, de grands volumes d'informations et de divers services. Espace virtuel, il devrait voir, à la différence du Web que nous connaissons aujourd'hui, les utilisateurs déchargés d'une bonne partie de leurs tâches de recherche, de construction et de combinaison des résultats, grâce aux capacités accrues des agents logiciels, en termes d'accès aux contenus des ressources et de raisonnement sur celles-ci [145].

Pour mener à bien cet objectif, certaines conditions doivent être remplies. Dans un premier temps, à chaque ressource sur le Web devrait être associée une représentation bien formalisée de son contenu informationnel, ainsi que des méta-données (*i.e.* son créateur, sa localisation, *etc.*). Dans un deuxième temps, il faut définir les moyens d'exploiter les annotations attachées aux ressources Web, à travers l'utilisation de techniques de recherche/extraction d'informations et d'inférence.

Ainsi, le Web Sémantique est d'abord une infrastructure pour permettre l'utilisation de connaissances formalisées, aidant la structuration du contenu informel actuel du Web, même si aucun consensus n'existe qui détermine jusqu'où cette formalisation doit aller [145]. Cette infrastructure doit permettre d'abord de localiser, d'identifier et de transformer des ressources de manière robuste et saine, tout en renforçant l'esprit d'ouverture du Web avec sa diversité d'utilisateurs. Elle doit s'appuyer sur un certain niveau de consensus portant, par exemple, sur les langages de représentation ou sur les ontologies utilisées [145]. Elle doit contribuer à assurer, le plus automatiquement possible, l'interopérabilité et les transformations entre les différents formalismes et les différentes ontologies. Ensuite, l'infrastructure du Web Sémantique doit faciliter la mise en oeuvre de calculs et de raisonnements complexes tout en offrant des garanties supérieures sur leur validité. Elle doit également offrir des mécanismes de protection (droits d'accès, d'utilisation et de reproduction), ainsi que des mécanismes permettant de qualifier les connaissances afin d'augmenter le niveau de

[1] Le rappel et la précision sont des critères de mesure des performances d'un moteur de recherche.

confiance des utilisateurs [145].

Mais, réduire le Web Sémantique à cette infrastructure serait, selon [145], trop limitatif, car les applications développées au-dessus de cette infrastructure contribuent pleinement au Web Sémantique en cela que c'est par le biais de leur utilisation que se caractérise le Web de demain.

3.2.1 Architecture

Les travaux œuvrant à la réalisation du Web Sémantique se situent à des niveaux de complexité très différents. Les plus simples utilisent des jeux plus ou moins réduits de méta-données dans un contexte de recherche d'information, ou pour adapter la présentation des informations aux utilisateurs. Dans ce cas, des langages de représentation simples sont suffisants. Dans les travaux plus complexes mettant en oeuvre des architectures sophistiquées pour permettre, par exemple, l'exploitation de ressources hétérogènes, des langages plus expressifs et plus formels issus des travaux en représentation et en ingéniérie des connaissances, sont nécessaires [145].

La proposition du W3C s'appuie au départ sur une pyramide de langages dont seules les couches basses (*i.e.* les couches URI, XML, RDF, RDF(S), OWL representées dans la figure 3.2) sont aujourd'hui relativement stabilisées [145]. La figure 3.2 montre l'organisation en couches du Web Sémantique, proposée par le W3C [33]. Deux types de bénéfice peuvent être attendus de cette organisation. (1) Elle permet une approche graduelle dans les processus de standardisation et d'acceptation par les utilisateurs. (2) Par ailleurs, si elle est bien conçue, elle doit permettre de disposer de langages à différents niveaux de complexité, en fonction de l'application à réaliser [145].

Un aspect central de l'infrastructure est sa capacité d'identification et de localisation des diverses ressources. Elle repose sur la notion d'URI (Uniform Resource Identifier) [4] qui, outre le fait d'identifier une ressource sur un réseau, fournit les moyens d'agir sur une ressource ou d'obtenir une représentation de la ressource en décrivant son mode d'accès primaire ou "emplacement" réseau. Par exemple, l'URL http ://www.wikipedia.org/ est un URI qui identifie une ressource (page d'accueil Wikipédia) et implique qu'une représentation de cette ressource (lien HTML à la page d'accueil, en caractères encodés) peut être obtenue via HTTP depuis un réseau hôte appelé www.wikipedia.org. Cette notion est à la base même des langages proposés par le W3C [145].

Une autre caractéristique de tous ces langages est d'être systématiquement exprimables et échangeables dans une syntaxe XML [43]. Ceci permet de bénéficier de l'ensemble des technologies développées autour d'XML : XML Schemas [51], outils d'exploitation des ressources XML (bibliothèques JAVA, etc.), bases de données gérant des fichiers XML, même si des langages de requêtes spécifiques [105] sont nécessaires pour les langages construits sur XML comme RDF.

Le premier des langages sémantiques est RDF auquel s'est ajouté rapidement RDF Schema (décrits dans la section 3.2.2.1). Les objectifs initiaux de RDF étaient la représentation et une meilleure exploitation des méta-données [145]. De manière générale, RDF permet de voir le Web comme un ensemble de ressources reliées par les liens étiquetés *sémantiquement*. Ainsi, XML peut être vu comme la couche de transport syntaxique, tandis que RDF peut être considéré comme un langage relationnel de base [145]. RDFS offre des primitives de représentation de structures ou primitives ontologiques.

La couche suivante évoquée par le W3C (voir figure 3.2) propose un langage standard pour la représentation et l'utilisation d'ontologies, appelé OWL [165], permettant la mise en œuvre d'un certain nombre d'inférences [145]. OWL découle de travaux de recherche sur les Logiques de Descriptions, sur les Langages de Frames, et sur les Réseaux Sémantiques, et offre une grande

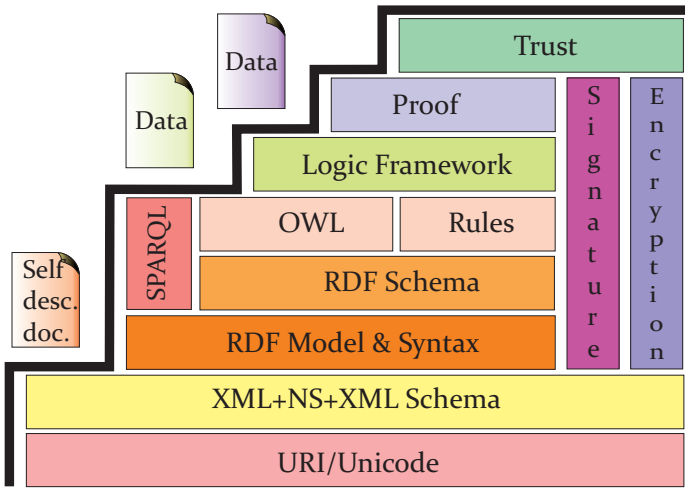


FIG. *3.2: L'architecture multi-couches du Web sémantique selon le W3C.*

expressivité ainsi qu'une palette étendue de raisonneurs (Pellet, RacerPro, Fact, Cerebra...) (voir section 3.2.4).

Dans une vision plus prospective, la pyramide du W3C inclut aussi des couches supérieures (*i.e. rules, proof, trust, signature, encryption*). La possibilité de faire des déductions plus complètes pourra s'appuyer sur la standardisation d'un langage de règles, comme RuleML (Rule Markup Language). De manière plus ambitieuse encore, selon Laubert *et al.* [145], la capacité de produire des preuves de déductions faites pourra augmenter le niveau de confiance des utilisateurs en ces déductions. Le problème de confiance doit reposer aussi sur des méthodes de qualification de l'origine de l'information, par exemple par des méta-données et des annotations, éventuellement certifiées par des signatures électroniques [145].

3.2.2 Langages

Le principe minimaliste impose que la construction du Web Sémantique soit basée sur un modèle de représentation d'une grande généralité, partagé par tous les acteurs. Ce modèle d'assertions sémantiques simples est concrétisé dans la recommandation RDF, proposée par le consortium W3C [44]. Le langage RDF a été créé pour représenter l'information de la manière la moins contraignante et la plus flexible possible [140].

3.2.2.1 Le langage RDF(S)

Le *Resource Description Framework* (RDF) [162], est le premier langage proposé par le W3C comme standard pour la description des ressources Web. RDF repose sur un modèle basé sur les concepts d'*assertion* et de *citation*[2]. Les assertions sont exprimées à l'aide de triplets de type ⟨*sujet, prédicat, objet*⟩, et peuvent être regroupées dans des graphes RDF[3]. Un exemple d'un tel graphe est illustré dans la figure 3.3.

L'écriture d'un triplet RDF indique qu'une relation, *prédicat*, existe entre les ressources représentées par le *sujet* et l'*objet* du triplet. Un triplet RDF est sémantiquement interprété comme

[2] Une *citation* est une assertion à propos d'une autre assertion.

[3] Un graphe RDF est défini comme un ensemble de triplets.

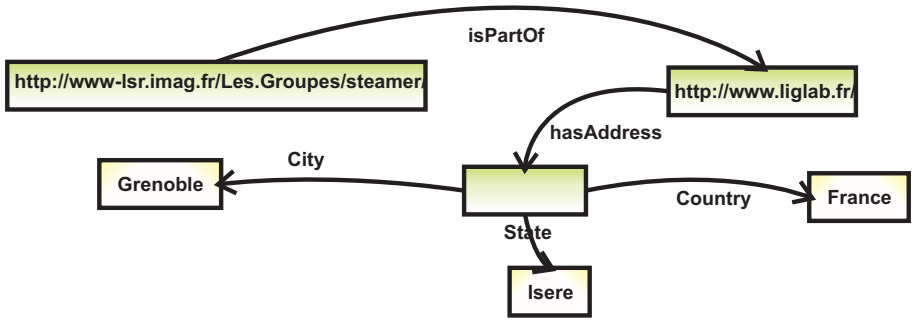


FIG. *3.3: Exemple simple de graphe RDF.*

l'assertion de la propriété, *prédicat P*, qui est satisfaite par la paire de ressources *sujet, S, objet, O,* : $P(S\ O)$. Le *sujet* représente, soit une référence URI (*Uniform Resource Identifier*), soit un nœud anonyme. Les URI sont utilisés pour identifier de façon unique des ressources Web dans le sens général du terme. Ils ne désignent pas forcément des ressources accessibles à travers un réseau, telles qu'un document électronique, une image, un service, etc., mais peuvent aussi référencer n'importe quel objet physique ou abstrait identifiable (un livre, une personne, une maladie, etc.). Les nœuds anonymes sont assimilables à des variables locales, au sens où elles sont utilisées pour désigner des ressources qui ne disposent pas d'un identifiant URI. Elles n'ont pas de nom intrinsèque, ni d'identifiant qui soit utilisable à l'extérieur du graphe RDF qui les définit. Elles ne peuvent pas désigner des valeurs concrètes (*littéraux*).

Le *prédicat* d'une assertion représente un aspect spécifique, une caractéristique, un attribut ou une relation utilisée pour décrire la ressource désignée par le *sujet.* L'*objet* représente la valeur de la propriété *prédicat* pour la ressource *sujet.* Il peut désigner l'URI d'une ressource ou bien une valeur concrète spécifiée à l'aide d'un littéral.

Par exemple, le graphe RDF illustré dans la figure 3.3 décrit l'équipe STEAMER, identifié par l'URI *http ://www-lsr.imag.fr/Les.Groupes/steamer/,* comme faisant partie du laboratoire LIG dont l'URI est *http ://www.liglab.fr/.* L'adresse du laboratoire LIG est modélisée à l'aide d'un noeud anonyme, pour lequel trois caractéristiques sont exprimées : le nom de la ville (Grenoble), le nom du département (Isère), et le nom du pays (France). La figure 3.4 reprend les mêmes descriptions, cette fois représentées en utilisant la syntaxe RDF/XML.

Expressivité et inférences

L'une des contraintes les plus importantes imposées au langage RDF est la généralité qui fait qu'il est peu expressif en lui-même et ne permet pas la définition de descriptions complexes. Comparé aux *Logiques de Description,* par exemple, le langage ne possède pas de négation, ni d'implication logique, il est donc très limité par rapport à son successeur, le langage OWL, présenté dans la section 3.2.2.3. Si on considère un ensemble de faits, il est facile de vérifier l'existence d'une preuve pour une question donnée, parce que ni les faits, ni les questions, n'ont assez d'expressivité pour poser des problèmes de décidabilité des raisonnements. Même si les documents RDF n'ont pas une expressivité importante, et même si le travail de traduction d'un modèle de données d'une application vers le modèle RDF n'est pas aisé, on peut toutefois trouver un intérêt dans son utilisation. En effet, les descriptions RDF, limitées et simples à l'intérieur d'une application, peuvent, grâce à leur sémantique formelle, être combinées avec des données issues d'autres

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">

<rdf:Description rdf:about=”http://www-lsr.imag.fr/Les.Groupes/steamer/”>
   <isPartOf rdf:ressource="http:/www.liglab.fr">
</rdf:Description>

<rdf:Description rdf:about=”http:/www.liglab.fr”>
   <hasAddress rdf:nodeID="abc">
</rdf:Description>

<rdf:Description rdf:nodeID="abc" City=”Grenoble”/>
<State=”Isere”>
<Country=”France”>

</rdf:Description>
</rdf:RDF>
```

FIG. *3.4: Sérialisation RDF/XML du graphe RDF de la figure 3.3.*

applications Web. L'existence d'un langage sémantique *élémentaire* et de *référence* qui garantisse l'interopérabilité est essentielle [32]. C'est le rôle joué par RDF Schéma.

RDF Schéma

Un grand avantage de RDF vient de son extensibilité, à travers l'utilisation de schémas RDF ou RDF(S) [44]. Ces derniers, construits à partir du fragment de la *Logique du Premier Ordre* illustré dans le tableau 3.1, peuvent être vus comme des méta-modèles qui spécifient quels sont les types de ressources qui peuvent être utilisés dans les triplets RDF, et leur associent une sémantique formelle. Ainsi, ils permettent la définition de *classes* d'objets et de *propriétés*, les deux organisées dans des hiérarchies de *subsomption*[4] [144]. Pour les propriétés, RDF(S) ajoute les notions de *"domaine"* et d'*"image"*, correspondant respectivement au domaine de définition et à l'ensemble d'arrivée d'une fonction. Ces méta-assertions permettent de réaliser des vérifications élémentaires sur le contenu des documents RDF, en testant, par exemple, si le sujet/objet d'un triplet est cohérent par rapport au domaine/image de la propriété qui le caractérise.

Constructeur	Axiomatisation LPO
La relation de subsomption entre *Classes*	$\forall X(C_1(X) \Rightarrow C_2(X))$
La relation de subsomption entre *Propriétés*	$\forall X, \forall Y(P_1(X,Y) \Rightarrow P_2(X,Y))$
Le domaine d'une Propriété	$\forall X, \forall Y(P_1(X,Y) \Rightarrow C(X))$
L'image d'une Propriété	$\forall X, \forall Y(P_1(X,Y) \Rightarrow C(Y))$

TAB. *3.1: Le fragment de la Logique de Premier Ordre sur lequel RDF(S) est basé.*

Les déclarations RDF définissent des relations entre des objets (nœuds d'un graphe) qui appartiennent à un univers sémantique (Schéma RDF). À chacun de ces univers sémantiques correspond un domaine nominal, identifié par un préfixe particulier, appelé *espace de noms*. L'usage de ces

[4] La *subsomption* désigne une relation hiérarchique entre des concepts, dans les logiques de description. Cette notion est proche de la relation “est impliqué par” en logique classique, ou encore “contient” en logique ensembliste.

derniers permet d'éviter toute confusion entre des définitions indépendantes - et peut-être conflictuelles - d'un même terme. Grâce aux espaces de noms, les schémas RDF peuvent s'intégrer et ne s'excluent pas mutuellement.

Interprétation

La sémantique du langage RDF(S) est une sémantique des "mondes possibles". L'intuition de base de la sémantique des modèles est qu'affirmer une phrase établit une revendication à propos du monde : c'est une autre façon de dire que le monde est, en fait, arrangé selon une interprétation qui rend la phrase vraie. En d'autre termes, une affirmation revient à énoncer une contrainte sur les états possibles du monde. Il est important de remarquer que l'on ne présume pas ici qu'une affirmation contienne suffisamment d'information pour indiquer une seule et unique interprétation. Il est généralement impossible de produire suffisamment d'affirmations dans un langage pour contraindre entièrement les interprétations à un seul monde possible, et donc, il n'existe pas une seule et unique interprétation d'un graphe RDF. En général, plus grand est le graphe RDF, plus il exprime de choses à propos du monde − plus petit est l'ensemble des interprétations vraies pour une assertion du graphe − moins il y a d'états possibles du monde qui vérifient le graphe affirmé.

Définition 3.1. *Formellement, une interprétation simple I d'un vocabulaire RDF, V, est composée de :*

- *un ensemble non-vide de ressources, IR, appelé le domaine ou l'univers de l'interprétation,*
- *un sous-ensemble distingué de IR, appelé IP, qui représente l'ensemble des propriétés de I ; un élément v du vocabulaire V est une propriété si et seulement si* $I(v) \in IP$,
- *une fonction d'appariement, IEXT, définie sur IP et avec des valeurs dans* $2^{IR \times IR}$,
- *une fonction d'appariement, IS, définie pour les références URI de V et avec des valeurs dans IR,*
- *un sous-ensemble distinctif LV de IR, appelé l'ensemble des valeurs littérales, qui contient tous les littéraux ordinaires dans V,*
- *une fonction d'appariement, IL, qui traduit les littéraux typés de V vers des éléments de IR.*

RDF impose également des restrictions sémantiques sur les interprétations, telles que : x est un élément de IP si et seulement si $< x, I(rdf : Property) > \in IEXT(I(rdf : type))$ [127].

RDFS ajoute des conditions sémantiques qui donnent du sens au vocabulaire fourni par le Schéma [127]. Par exemple :

- l'ensemble de toutes les classes dans une interprétation sera nommé IC : $IC = ICEXT(I(rdfs : Class))$. L'application $ICEXT$ (abréviation de l'Extension de Classe dans I) définie de IC vers l'ensemble des sous-ensembles de IR, fournit une interprétation extensionelle pour les classes.
- une classe est interprétée comme l'ensemble d'objets dont le type est la classe respective : $x \in ICEXT(y) \Leftrightarrow \langle x,y \rangle \in IEXT(I(rdf : type))$.
- si $\langle x,y \rangle \in IEXT(I(rdfs : subClassOf))$ alors $x,y \in IC$ et $ICEXT(x) \subseteq ICEXT(y)$,
- $IEXT(I(rdfs : subClassOf))$ est transitif et réflexif sur IC,
- $IEXT(I(rdfs : subPropertyOf))$ est transitif et réflexif sur IP,
- si $\langle x,y \rangle \in IEXT(I(rdfs : domain))$ et $\langle u,v \rangle \in IEXT(x)$, alors $u \in ICEXT(y)$
- ...

Des explications plus complètes et détaillées sur l'interprétation des graphes RDF(S) sont présentées dans [127].

Limitations

Le langage RDF, même étendu par les schémas RDF, se révèle souvent insuffisamment expressif et comporte un certain nombre de limites. On peut citer, par exemple, l'absence d'axiomes[5], qui forment le support principal d'inférences. Ainsi, en RDF(S) on ne peut pas spécifier deux classes comme étant disjointes, et on ne peut pas non plus construire de nouvelles classes par combinaison ensembliste (intersection, union, complément,...) de classes existantes. Également, les propriétés RDF(S) ne peuvent pas être contraintes en spécifiant des restrictions sur le nombre d'occurrences de valeurs qu'elles peuvent prendre. Leur comportement ne peut pas être défini à l'aide de caractéristiques telles que la transitivité (*i.e.* `estPlusGrandQue`), la propriété inverse (*i.e.* `mange` est la propriété inverse `estMangéPar`), *etc.*

Une autre limitation importante provient du fait que RDF(S) ne permet pas de réaliser des appariements entre des graphes RDF(S) différents, afin de spécifier des correspondances entre des classes, propriétés ou individus issus d'espaces de noms différents, mais pouvant être considérés comme équivalents. Cela constitue un grand obstacle pour l'intégration de graphes RDF(S) spécifiant des domaines qui se recouvrent.

Concernant la modélisation des informations spatiales, qui nous intéresse particulièrement dans cette thèse, elle ne peut pas se faire en utilisant des types spatiaux dédiés, car ils ne sont pas supportés par le Schéma XML. Néanmoins, l'information spatiale peut être décrite à travers l'utilisation de concepts spatiaux, modélisés en utilisant les schémas RDF. On notera toutefois que ce type de modélisation n'est pas du tout aisé ni intuitif, et reste très compliqué à comprendre par un utilisateur non expert.

L'information temporelle est, quant à elle, plus facile à représenter grâce aux types temporels proposés par le Schéma XML : `xsd:dateTime`, `xsd:time`, `xsd:date`, `xsd:gYearMonth`, `xsd:gYear`,
`xsd:gMonthDay`, `xsd:gDay`, `xsd:gMonth` [127]. Néanmoins, si l'on peut attacher une validité temporelle aux *classes* et aux *individus*, il est impossible de faire de même pour les *propriétés*. Le besoin est cependant très fort si on pense que la plus grande partie des relations sont inscrites dans un contexte temporel. Par exemple, la relation `travailleAvec` a un début et une fin, dans la majorité des cas, très bien définis.

Les schémas RDF possèdent des primitives puissantes (`rdfs:Class` - la classe de toutes les classes et `rdfs:Property` - la classe de toutes les propriétés) [165], qui offrent un support pour la méta-modélisation. Cependant, une meilleure expressivité obtenue par une simple extension de RDF Schéma peut avoir comme conséquence l'indécidabilité des raisonnements (*i.e.* OWL Full est obtenu comme extension de RDF(S)).

Pour résumer, parmi les grandes limitations de RDF(S), on compte l'absence de notion de modularité, la portée globale plutôt que locale des propriétés, l'impossibilité de déclarer des classes disjointes, des classes complexes à l'aide d'expressions ensemblistes, et des restrictions de cardinalité. Ces inconvénients disparaissent dans des langages plus expressifs, construits au-dessus RDF(S), comme OWL ou son extension OWL 2. Ceux-ci organisent les connaissances autour de la notion d'ontologie, présentée dans la section suivante.

[5] Une axiome est une proposition considérée comme vraie.

3.2.2.2 Ontologie - définitions et utilité

Le terme *ontologie* est employé dans des contextes très différents touchant la philosophie, la linguistique ou l'intelligence artificielle [71]. L'ontologie en tant que discipline, est la partie de la philosophie qui s'intéresse à la nature et à l'organisation *a priori* de la réalité [119]. Ici, l'*Ontologie* essaie de trouver des réponses à la question "*Qu'est ce qu'un être ?* " ou bien, en d'autre mots : "*Quels sont les traits communs à tous les êtres* ?" [118].

En informatique, la communauté Ingénierie des Connaissances a commencé à utiliser le terme *ontologie* au début des années 90, lorsque celles-ci ont été conçues pour assister l'acquisition de connaissances dans le cadre des systèmes à base de connaissances (SBC) [56]. Dans ces domaines, ontologie est souvent synonyme de modèle conceptuel, ce qui est assez éloigné du sens philosophique. Welty introduit cependant une nuance [237]. Il définit un modèle conceptuel comme l'implémentation d'une ontologie qui satisfait les contraintes d'une application. En revanche, une ontologie est indépendante des contraintes d'exécution, son but étant de spécifier la conceptualisation du monde sous-jacent à l'application [237]. Il rejoint la première définition de Gruber, pour qui une ontologie est la spécification explicite d'une conceptualisation [114].

Dans ce domaine, les ontologies suivent les principes fondateurs en matière de représentation de connaissances, qui séparent une base de connaissances "*déclarative*" et un moteur d'inférence "*procédural*", autrement dit qui séparent les connaissances du domaine modélisé et les mécanismes d'exploitation de ces connaissances. Le but de cette séparation modulaire est aussi de construire des SBC plus facilement et plus rapidement, en réutilisant le plus de composants génériques possible, que ce soit au niveau du raisonnement ou des connaissances du domaine.

À ce jour, plusieurs définitions ont été proposées pour les ontologies informatiques [13, 130, 26, 64, 139, 118, 113]. Parmi celles-ci, nous reprenons la définition proposée par Cocchiarella dans son livre *Formal Ontology* [64], qui désigne une ontologie formelle comme étant "*le développement systématique, formel et axiomatique de la logique de tous les formes et modes d'existence*[6]". Comme le souligne Guarino dans [118], cette définition prend en compte les deux acceptations de l'adjectif *formel* : d'un coté la *rigueur* des descriptions, et, de l'autre, la *structure* des choses. Concrètement, on ne s'intéresse pas seulement à l'existence de certains individus, mais plutôt à la description formelle de leurs traits structurels. C'est donc une théorie des distinctions *a priori* entre les entités du monde et entre les catégories utilisées pour modéliser le monde [119].

Gruber, quant à lui, perçoit l'ontologie comme une description explicite d'une conceptualisation [113], où la notion de conceptualisation représente une vue abstraite et simplifiée du monde que l'on veut représenter pour un besoin précis. Toujours selon Gruber, ce qui *existe* est ce qui peut être *représenté*. Lorsque les connaissances d'un domaine sont représentées en utilisant un formalisme déclaratif, l'ensemble des objets qui peuvent être définis forment l'*univers du discours*. Cet ensemble contenant les objets généralement reconnus comme existant dans le domaine, leurs définitions et les axiomes qui les lient se reflètent alors dans le *vocabulaire* ou dans l'*ontologie* du domaine. En d'autres termes, une ontologie est une spécification explicite d'une modélisation conceptuelle d'un domaine du monde réel, qui met en jeu des concepts et les relations qui les lient ou une représentation de cette modélisation [113]. Ainsi, construire une ontologie d'un domaine revient à décider *quels* objets existent (sont à représenter), quelles sont les propriétés de ces objets et quelles relations sont établies entre eux.

Pour Charlet *et. al* [56] ainsi que pour Horrocks *et. al* [135], les ontologies sont au centre

[6] En anglais :"*systematic, formal, axiomatic development of the logic of all forms and modes of being*"

des développements visant le Web Sémantique. Ce dernier s'appuie sur l'annotation de ressources Web à l'aide de concepts formels afin de permettre à des logiciels d'interpréter de façon non ambigüe la sémantique de ces annotations pour réaliser des inférences. Dans ce cadre, les ontologies sont utilisées pour spécifier un vocabulaire complexe, structurer et exploiter des méta-données, pour intégrer des sources de données hétérogènes, pour décrire les services Web, et, en général, partout où il est nécessaire d'appuyer le travail informatique sur des représentations sémantiques réclamant un certain consensus [56]. Il est important de souligner que, contrairement à la validité d'une ontologie philosophique qui est absolue, celle d'une ontologie informatique dépend d'un consensus, elle est donc plus restreinte. Ce point amène Gruber à proposer une seconde définition d'une ontologie, comme un accord sur une conceptualisation partagée [114]. Il peut alors exister plusieurs ontologies concurrentes du même domaine. Guarino ajoute que la conceptualisation peut n'être que partielle [117].

En termes d'applications, comme l'illustrent Horrocks *et. al* [135], les ontologies peuvent être utilisées notamment :

- dans le commerce électronique, pour faciliter la communication entre les agents qui vendent et ceux qui achètent des biens et/ou des services, décrits dans les termes d'un vocabulaire commun formalisé à l'aide des ontologies ;
- pour la découverte de services Web spécialisés, car elles peuvent fournir des descriptions riches et non ambigües ;
- pour la construction de moteurs de recherche sémantiques, capables de retrouver les pages Web contenant des mots ou des phrases, similaires du point de vue sémantique mais différentes du point de vue syntaxique.

Actuellement, l'un des résultats les plus importants qui découlent des travaux réalisés sur le Web Sémantique et les ontologies, est le langage OWL, devenu un standard pour la définition des ontologies, et offrant différents niveaux d'expressivité.

3.2.2.3 OWL, le langage standard pour la définition d'ontologies

Afin de permettre une recherche performante et un partage efficient des connaissances sur le Web, on annote les pages en décrivant leur contenu à l'aide d'informations qui peuvent être facilement comprises par les agents logiciels. Ces annotations sémantiques sont construites à partir d'un vocabulaire commun et formalisé, qui assure une compréhension unique et partagée des déclarations, entre les différents acteurs [186]. Pour cela, il faut faire appel à des ontologies, constituées de collections de définitions de concepts et de relations, partagées par les différents agents et servant de référence.

Le langage standard proposé par le consortium W3C [73] pour la représentation d'ontologies s'appelle OWL (Web Ontology Language). Il a été imaginé comme une extension du langage RDF (décrit dans la section 3.2.2.1), dans le sens où, pour définir les domaines d'application, OWL utilise les primitives de base modélisées par les schémas RDF. OWL permet donc de définir des *classes* qui représentent des ensembles d'*individus*[7], et des *propriétés* qui représentent les caractéristiques des *individus* ou les relations possibles entre ces derniers. Les connaissances sont prises en compte selon deux niveaux d'abstraction : la représentation des concepts et des relations relève du niveau *terminologique* (TBox), tandis que la représentation des *individus* et les assertions

[7] Dans ce document, nous utilisons les termes *objet* et *individu* indifféremment et de façon interchangeable pour désigner l'*instance* d'une *classe*.

dans lesquelles ils interviennent relèvent du niveau *assertionnel* (ABox) (voir pages 59 et 60).

Les *classes* et les *propriétés* sont organisées dans des hiérarchies de subsomption à l'aide, respectivement, des relations `rdfs:subClassOf` et `rdfs:subPropertyOf` [29]. Pour les *propriétés*, OWL permet d'en spécifier le *domaine* et *l'image*. Les domaines des *propriétés* OWL sont des *classes* OWL, tandis que les images peuvent être, soit des *classes* OWL, soit des *types de données* comme les *entiers*, les *chaînes de caractères*, *etc.* [29].

Constructeur OWL	Syntaxe LD	Exemple
`intersectionOf`	$C_1 \sqcap C_2$	$Personne \sqcap Masculin$
`unionOf`	$C_1 \sqcup C_2$	$Docteur \sqcup Professeur$
`complementOf`	$\neg C$	$\neg Homme$
`oneOf`	$\{x_1, x_2, \ldots, x_n\}$	$\{Pierre, Paul, Maria\}$
`allValuesFrom`	$\forall r.C$	$\forall aEnfant.Homme$
`someValuesFrom`	$\exists r.C$	$\exists aEnfant.Homme$
`hasValue`	$\exists r.\{x\}$	$\exists citoyenDe.\{Europe\}$
`minCardinality`	$\geq n.r$	$\geq 2\ aEnfant$
`maxCardinality`	$\leq n.r$	$\leq 1\ aEnfant$
`inverseOf`	r^-	$aEnfant^-$

TAB. *3.2: Correspondances entre les constructeurs de OWL(DL) et les Logiques de Description (d'après [186]).*

Aux classes et aux propriétés, OWL ajoute d'autres structures de représentation dérivées des Logiques de Description qui augmentent son expressivité. Ainsi, les *classes* OWL peuvent être spécifiées comme des combinaisons logiques (intersection, union ou complément) d'autres *classes*, ou bien par énumération explicite de leurs *objets* (voir le tableau 3.2). OWL offre la possibilité de spécifier pour un *objet* donné, l'ensemble des *classes* auxquelles il appartient et les valeurs des *propriétés* qui lui sont attachées. Des axiomes spécifiques (`owl:equivalentClass`, `owl:equivalentProperty`) [29] ont été introduits, qui traduisent la définition d'équivalences respectivement entre *classes* et entre *propriétés* (voir le tableau 3.3 qui présente les axiomes de OWL DL). Les *classes* OWL peuvent aussi être définies comme disjointes en utilisant l'axiome `owl:disjointWith`, tandis que les individus peuvent être déclarés comme égaux ou, au contraire, inégaux à l'aide des axiomes `owl:sameAs` et `owl:differentFrom` [29]. OWL permet également de définir quatre types de caractéristiques pour les propriétés : fonctionnalité, inverse-fonctionnalité, transitivité, symétrie.

L'avantage le plus net de OWL, par rapport à d'autres langages à ontologies comme RDF(S) [44] est, selon [135], la possibilité de restreindre le comportement local des *propriétés*. Ainsi, il est possible de spécifier une *classe* OWL pour laquelle une *propriété* particulière *p*, est contrainte de manière que :

- toutes ses valeurs appartiennent à une autre *classe* ou *type de données* ;
- il existe au moins une valeur de la *propriété p* qui soit incluse dans l'extension d'une *classe* ou d'un *type de donnée* spécifié ;
- la propriété ait au moins certaines valeurs prédéfinies ;
- la propriété ait au moins/au plus un certain nombre de valeurs distinctes.

Axiome OWL	Syntaxe LD	Exemple
`subClassOf`	$C_1 \sqsubseteq C_2$	$Homme \sqsubseteq Personne$
`equivalentClass`	$C_1 \equiv C_2$	$Homme \equiv Masculin \sqcap Personne$
`subPropertyOf`	$r_1 \sqsubseteq r_2$	$aFils \sqsubseteq aEnfant$
`equivalentProperty`	$r_1 \equiv r_2$	$cout \equiv prix$
`disjointWith`	$C_1 \sqsubseteq \neg C2$	$\{Masculin\} \sqsubseteq \neg\{Feminin\}$
`sameAs`	$\{x_1\} \equiv \{x_2\}$	$\{Paris\} \equiv \{Parigi\}$
`differentFrom`	$\{x_1\} \sqsubseteq \neg\{x_2\}$	$\{Paolo\} \sqsubseteq \neg\{Maria\}$
`transitiveProperty`	$r \in R_+$	$aAncetre^+ \in R_+$
`functionalProperty`	$\top \sqsubseteq (\leq 1\ r)$	$\top \sqsubseteq (\leq 1\ aMaman)$
`inverseFunctionalProperty`	$\top \sqsubseteq (\leq 1\ r^-)$	$\top \sqsubseteq (\leq 1\ estMamanDe^-)$
`symmetricProperty`	$r \equiv r^-$	$estVoisinDe \equiv estVoisinDe^-$

TAB. *3.3: Correspondances entre les axiomes de OWL(DL) et les Logiques de Description (d'après [186]).*

Expressivité croissante : De OWL Lite à OWL Full

Pour répondre à des besoins de représentation et d'inférence variés, exprimés à travers différentes applications, le consortium W3C propose trois sous langages : OWL Lite, OWL DL et OWL Full, énumérés ici par ordre décroissant d'expressivité.

OWL Full [225] est la version la plus complexe de OWL, celle qui offre le plus haut niveau d'expressivité. Elle autorise l'utilisation de toutes les primitives OWL combinées d'une façon arbitraire aux structures RDF et RDF Schema. Ceci inclut la possibilité de changer le sens prédéfini des structures, en mélangeant modèle *extensionel* et *intensionel* par le biais de ce que Motik appelle méta-modélisation [178]. Par exemple, on pourrait définir une restriction pour la classe de toutes les classes qui impose une limitation du nombre de classes qui peuvent appartenir à une même ontologie. Également, on peut simultanément traiter une classe comme étant une collection d'individus ou un individu à part entière. Les avantages de OWL Full sont une compatibilité totale avec RDF, du point de vue syntaxique et sémantique, et l'expressivité maximale qu'il assure. En contre-partie, sa grande limitation est l'indécidabilité des raisonnements (*i.e.* vérification de la cohérence, test de subsomption, *etc.*). Il est ainsi peu probable qu'un système de raisonnement puisse mettre en œuvre toutes les caractéristiques de OWL Full [167, 178].

OWL DL s'adresse aux utilisateurs qui souhaitent une expressivité maximale sans perdre la complétude et la décidabilité du calcul de *subsomption*. Pour rendre les raisonnements décidables, le sous-langage OWL DL impose des restrictions vis-à-vis de l'utilisation des constructeurs d'OWL et de RDF pour que le langage corresponde de près à la Logique de Description $\mathcal{SHIONQ}(\mathcal{D})$ [186]. Celle-ci est une extension de la Logique de Description $\mathcal{SHIQ}$, qui regroupe l'ensemble des constructeurs $\mathcal{ALC}$ (les concepts atomiques ($\mathcal{AL}$), la quantification existentielle ($\mathcal{E}$) et l'union ($\mathcal{U}$)) noté $\mathcal{S}$ auxquels est ajouté l'ensemble $\{\geq n\ r.C, \leq n\ r.C\}$, noté $\mathcal{Q}$. $\mathcal{H}$ exprime l'existence d'une hiérarchie de rôles (en OWL, les rôles sont remplacés par les propriétés) et $\mathcal{I}$ la possibilité de définir des rôles inverses et transitifs. $\mathcal{SHIQ}$ peut être étendue en $\mathcal{SHOQ}$ avec les types énumérés (noté $\mathcal{O}$) et en $\mathcal{SHIONQ}(\mathcal{D})$ avec les types de données primitifs (tels que entiers, booléens, chaînes de caractères, *etc.*) [186]. Les correspondances qui existent entre les constructeurs de OWL DL et ceux retrouvés dans les Logiques de Description sont illustrées dans le tableau 3.2, qui contient aussi des exemples d'utilisation de ces derniers.

OWL Lite est le langage le plus simple (il est moins expressif que les deux autres), il est décidable et complet. Il s'adresse aux utilisateurs qui ont principalement besoin de hiérarchies de classification et de constructeurs élémentaires. Il s'appuie sur une restriction de $\mathcal{SHIQ}$, pour laquelle les rôles sont uniquement fonctionnels, les types énumérés sont éliminés, ainsi que les axiomes d'incompatibilité entre deux concepts. OWL Lite interdit l'utilisation de l'*union* et du *complément* lors de la définition d'une *classe* ainsi que la présence d'*individus* dans les descriptions de classes ou d'axiomes. Les restrictions de cardinalité sont également limitées à 0 ou 1. L'un des grands avantages de OWL Lite est qu'il est facile à apprendre et qu'il existe des procédures de raisonnement efficaces [186].

Types de données

OWL utilise les types de données définis par le Schéma XML [51], qui offre des types de base qui, même s'ils sont extensibles, ne peuvent pas être référencés depuis une ontologie. À cause de l'absence d'un moyen standard pour aller d'un appel d'adresse URI à un type de données défini par un Schéma XML, il n'existe pas non plus de moyen standard pour utiliser en OWL les types de données du Schéma XML définis par l'utilisateur [197]. Les types énumérés, ne sont pas assez expressifs pour représenter une alternative valide aux types définis par l'utilisateur, car ils ne permettent pas l'expression de types de données infinis comme, par exemple, les nombres entiers supérieurs à 20 [195].

Une autre limitation importante du système de types de OWL est l'absence de primitives pour la définition de contraintes sur plusieurs propriétés [195]. Par exemple, on ne peut pas imposer que la somme de la `hauteur`, de la `longueur` et de la `largeur` d'un objet petit (`PetitObjet`) soit inférieure ou égale à 15 cm. OWL n'intègre pas de tels prédicats n-aires et n'offre pas de support pour la définition de nouveaux prédicats.

De plus, OWL ne permet pas de définir des types de données à travers une négation. On peut prendre comme exemple le type *entiers sauf 0*, qui pourrait être défini comme la négation du type énuméré `oneOf(''0''xsd:integer)` [195]. Une autre limitation vient du fait que les types énumérés ne peuvent pas être nommés en OWL. L'utilisateur doit donc (re)construire le type énuméré chaque fois qu'il souhaite l'utiliser. Cela constitue des problèmes assez importants en termes de maintien de la cohérence de l'ontologie et par rapport aux mises à jour, surtout si le type visé comporte un nombre important d'éléments [195].

Sémantique formelle

Une sémantique formelle, inspirée de la *théorie des modèles*, est associée aux *classes*, *propriétés* et *individus* définis dans les ontologies OWL. Son but est de spécifier de façon non ambigüe les relations qui existent entre la syntaxe du langage et un modèle concret du domaine visé [197, 135]. Un modèle consiste principalement en un domaine d'interprétation, noté Δ^I, qui représente un ensemble d'*objets*, et une fonction d'interprétation $\bullet^I$, qui associe aux noms des *individus*, *classes* et *propriétés*, respectivement des éléments du domaine, des ensembles d'éléments du domaine, et des relations binaires entre des éléments du domaine. Par exemple, pour l'individu *Paul*, l'interprétation est un élément du domaine, $Paul^I \in \Delta^I$, tandis que l'interprétation de la classe *Personne* est un ensemble d'objets : $Personne^I \subseteq \Delta^I$, alors que l'interprétation de la propriété *ami* est $ami^I \subseteq Personne^I \times Personne^I \subseteq \Delta^I \times \Delta^I$.

L'intégration des types de données dans le langage OWL est, elle aussi, fortement influencée par les recherches autour des Logiques de Description, qui portent une attention particulière à l'interprétation des types de données afin d'éviter la montée en complexité et l'indécidabilité des raisonnements. Comme il est montré dans [135], ces objectifs peuvent être satisfaits par le biais d'une séparation stricte entre, d'un côté, le domaine d'interprétation des types de données et des valeurs et, de l'autre côté, celui des classes et des individus. En suivant l'exemple de $\mathcal{SHOQ}(\mathcal{D})$, OWL inclut un domaine d'interprétation pour les valeurs Δ_D^I, disjoint du domaine d'interprétation des individus Δ^I. Ainsi, les types de données, tels que entiers ou chaînes de caractères, sont interprétés comme des sous-ensembles de Δ_D^I, tandis que les valeurs comme "23" ou "Paul" sont interprétées comme des éléments de Δ_D^I. Cette séparation est poussée plus loin, car les propriétés sont elles aussi divisées, d'un côté en *propriétés de type lien*[8](ou encore *propriété-lien*), ayant comme domaine et image des classes, et, de l'autre, en *propriétés de type attribut*[9] (ou encore *attribut*), ayant comme image un type de données. Les *propriétés-lien* sont interprétées comme des relations binaires sur le domaine Δ^I tandis que les *attributs* sont interprétés comme des relations binaires entre Δ^I et Δ_D^I. L'avantage de cette division réside dans la séparation claire entre le raisonnement à base de *classes* et d'*individus* et celui à base de *types de données* et de *valeurs*. Ainsi, si la décidabilité du raisonnement à base de *valeurs* et de *types de données* est garantie, le langage est décidable [135, 158]. Comme le montre Lutz dans sa thèse [158], pour un ensemble usuel de types de données tels que entiers, réels et chaînes de caractères, les raisonnements respectent la contrainte de décidabilité.

La fonction d'interprétation traite de façon adaptée les *ontologies* et les *annotations*. Elle est également étendue pour gérer les descriptions des *classes* complexes créées en utilisant des expressions ensemblistes ou des restrictions de propriétés, ainsi que pour les axiomes et les assertions des faits [197].

Par la suite, nous rappelons très brièvement l'organisation d'une ontologie selon deux dimensions : la TBox et la ABox. Selon Pan [195], la structuration d'une ontologie est très proche de celle d'une base de connaissances telle que définie par les Logiques de Description. Concrètement, une ontologie est divisée en deux parties distinctes : la partie intensionelle ou TBox, qui contient des informations sur le domaine modélisé (*i.e.* définitions de classes et de propriétés), et la partie extensionnelle ou ABox, contenant des descriptions sur des situations concrètes, spécifiques (*i.e.* définition d'individus et de leurs propriétés) [195].

TBox. La TBox d'une ontologie est définie comme l'ensemble des descriptions portant sur les classes (*i.e.* la définition des relations de subsomption entre classes, la définition des équivalences entre classes, *etc.*), sur les propriétés (*i.e.* la définition des relations de subsomption entre propriétés, la définition des équivalences entre propriétés, la définition des propriétés transitives, inverses, symétriques *etc.*), et sur la façon dont les propriétés et les classes sont connectées entre elles [195]. Un exemple de description terminologique est $\exists hasStudent.PHDStudent \sqsubseteq AcademicStaff \sqcap FullTimeStaff$, traduite en OWL comme :

```
<owl:Class>
       <owl:intersectionOf rdf:parseType="Collection">
              <rdf:Description rdf:about="#AcademicStaff"/>
```

[8] En anglais *object property*.

[9] En anglais *datatype property*.

```
            <rdf:Description rdf:about="#FullTimeStaff"/>
        </owl:intersectionOf>
        <rdfs:subClassOf>
            <owl:Restriction>
                <owl:onProperty rdf:resource="#hasStudent"/>
                <owl:someValuesFrom rdf:resource="#PHDStudent"/>
            </owl:Restriction>
        </rdfs:subClassOf>
</owl:Class>
```

Cette dernière exprime le fait que seuls les académiques à plein temps peuvent encadrer des étudiants en thèse.

Les définitions incluses dans la TBox d'une ontologie constituent l'ensemble des *axiomes terminologiques* [195]. Ceux-ci sont interprétés en termes de théorie des modèles. Ainsi, une interprétation I satisfait :

- une inclusion de classes, $C \sqsubseteq D$, écrit comme $I \Vdash C \sqsubseteq D$, ssi $C^I \subseteq D^I$;
- une équivalence de classes, $C \equiv D$, écrit comme $I \Vdash C \equiv D$, ssi $C^I = D^I$;
- une inclusion de propriétés, $P \sqsubseteq Q$, écrit comme $I \Vdash P \sqsubseteq Q$, ssi $P^I \subseteq Q^I$;
- ...
- une TBox T ($I \Vdash T$) ssi I satisfait toutes les descriptions terminologiques incluses dans T.

ABox. La ABox d'une ontologie contient la description d'un état particulier du monde, spécifié à travers la définition d'individus, de leurs propriétés spécifiques, et des relations concrètes qui existent entre eux [195]. Les déclarations incluses dans la ABox s'appellent *assertions* ou *axiomes d'individus*. La fonction d'interprétation I est étendue pour prendre en compte les assertions.

Ainsi, l'interprétation I satisfait l'assertion d'un individu, exprimée dans la forme $a : C$ (en OWL `<C rdf:about="#a"/>`) ssi $a^I \in C^I$. On dit alors que $a : C$ est une conséquence logique de I ($I \Vdash a : C$). L'interprétation I satisfait l'assertion d'une propriété entre deux individus (ou entre un individu et une valeur concrète) $\langle a, b \rangle : R$ ssi $\langle a^I, b^I \rangle \in R^I$. Une interprétation I satisfait une ABox A ($I \Vdash A$) si et seulement si I satisfait toutes les descriptions d'individus incluses dans A [195].

Raisonnements

Le standard OWL permet la modélisation d'ontologies, en imposant certaines règles syntaxiques et sémantiques. Des inférences peuvent être opérées à partir de *faits* (ABox) et d'*axiomes* (TBox) contenus dans les domaines modélisés en utilisant des *raisonneurs*[10] externes, en majorité basés sur les Logiques de Description. Les raisonnements à base de Logiques de Description (et implicitement ceux à base d'ontologies) consistent en un processus de découverte de connaissances implicites, impliquées[11] par l'ontologie considérée (*i.e.* la TBox et la ABox de l'ontologie sur laquelle on souhaite raisonner). Les principaux services d'inférence mis à disposition par les Logiques de Description sont :

- la vérification de la satisfiabilité d'une ontologie O, qui consiste à vérifier qu'il existe un modèle I de O, tel que $I \Vdash O$;
- la vérification de la satisfiabilité d'une classe C, qui consiste à vérifier qu'il existe un modèle I de O, tel que $I \Vdash C$;

[10] Un *raisonneur sémantique* ou *moteur de raisonnement* ou *moteur de règles* ou *raisonneur* est un logiciel capable de déduire des conséquences logiques à partir d'un ensemble connu de faits et/ou d'axiomes.

[11] En anglais *entailed*.

– le test de subsomption qui vise à vérifier que $C \sqsubseteq D$ est une conséquence logique de l'ontologie considérée, O, $:O \Vdash C \sqsubseteq D$. Concrètement, ce test consiste à prouver que l'inclusion $C^I \subseteq D^I$ est satisfaite dans tous les modèles I de O ;
– la vérification d'instance qui cherche à prouver que $a : C$ est une conséquence logique de l'ontologie considérée, O, $:O \Vdash a : C$. Ce test est équivalent à la vérification de l'inclusion $a^I \in C^I$ pour tous les modèles I de O.

Par ailleurs, le traitement des restrictions sur les cardinalités des classes ou propriétés, le test d'égalité entre chaînes de caractères ou la résolution des équations polynomiales sur des réels ou des cardinaux sont des raisonnements algébriques acceptés par certains raisonneurs. Une analyse des raisonneurs qui traitent les ontologies OWL est présentée dans la section 3.2.4.

Limitations

Très expressif et général, le langage OWL présente cependant un certain nombre de limitations de représentation. Parmi ces limitations, on trouve l'absence de structures adaptées pour la modélisation de relations n-aires, et pour la définition des différents rôles attachés aux objets impliqués dans des relations n-aires, l'absence de primitives pour exécuter du code (*i.e.* réflexe, attachement procédural), pour spécifier des valeurs par défaut, l'impossibilité de définir et d'utiliser des types complexes [186]. Il est aussi important de noter qu'il est difficile de travailler avec des valeurs concrètes (*nominals*), d'exprimer des contraintes lors de la définition d'un concept ou d'une relation, d'exprimer des comportements particuliers, *etc.*

3.2.2.4 Évolution OWL 2

Le langage OWL 2 est une extension ainsi qu'une révision du langage OWL, développé par le groupe de travail *W3C Web Ontology Working Group*, avec l'objectif de faciliter la création d'ontologies et leur partage via le Web [5]. Il vient compléter le langage OWL par un ensemble d'axiomes (dont l'ajout a été explicitement demandé par diverses communautés d'utilisateurs de OWL, afin de faciliter certaines descriptions terminologiques ou pour augmenter l'expressivité des ontologies), pour lesquels il existe à présent des algorithmes de raisonnement efficaces, et qui sont soutenus par les développeurs d'outils basés sur OWL [198]. L'objectif initial de OWL 2 a été d'exploiter les résultats récents de la recherche autour des Logiques de Description afin d'éliminer certaines des limites d'expressivité du langage OWL [110].

La structure des ontologies OWL 2 est précisée à l'aide de diagrammes UML [182], ce qui suggère un rapprochement entre les ontologies et le paradigme objet. En dehors de ce nouveau regard sur les ontologies, Motik *et al.* définissent également une nouvelle syntaxe pour l'encodage des ontologies, appelée *OWL 2 Functional-Style Syntax* [182, 181]. Celle-ci permet de définir des ontologies plus compactes, et qui sont plus faciles à lire et à éditer par les utilisateurs. La nouvelle syntaxe est similaire à la syntaxe abstraite de OWL, mais elle n'est cependant pas compatible avec celle-ci [182].

Du point de vue de l'organisation générale des ontologies, OWL 2 propose la séparation des structures de modélisation en trois catégories distinctes :

a) les **entités**, identifiées par des URI et considérées comme des termes primitifs, elles sont divisées à leur tour en :
   – classes (*i.e.* `Person`, `Student`, `Address`, `University`, *etc.*) ;

  - propriétés (*i.e.* `knows`, `livesAt`,*etc.*)
  - individus (*i.e.* `pers1`, `UJF`, `address1`, *etc.*)

b) les **expressions**, vues comme des notions complexes construites à partir des termes primitifs (voir l'exemple 3.1).

c) les **axiomes**, vus comme des déclarations considérées vraies dans le domaine décrit par l'ontologie qui les inclut (voir l'exemple 3.1).

Exemple 3.1. *Exemple de structures OWL 2 :*

Expressions :

```
IntersectionOf(a:Student a:Person)
InverseOf(a:livesAt)
```

Axiomes :

```
SubClassOf(a:Student, a:Person)
TransitiveProperty(a:ancestorOf)
EquivalentProperties(InverseOf(a:livesAt), a:isLocationOf)
```

Parmi les nouvelles fonctionnalités de représentation offertes par OWL 2, on peut citer le *sucre syntaxique*[12], quelques nouvelles restrictions de propriétés, le support étendu pour les types de données, des facilités simples de méta-modélisation, des annotations étendues [198]. Une description complète de l'extension OWL 2, est présentée dans [104], nous n'en donnons qu'un aperçu dans ce qui suit.

Le sucre syntaxique

OWL 2 offre deux structures de description qui n'augmentent pas l'expressivité du langage, mais qui rendent certaines descriptions plus faciles à définir. Il s'agit de `DisjointUnion`, axiome qui définit une classe comme l'union de deux ou plusieurs classes disjointes. Elle est équivalente à la combinaison entre les axiomes `owl:disjointWith` et `owl:equivalentClasse`.

La deuxième structure syntaxique proposée par OWL 2 est la négation d'une assertion de propriété, déclinée pour les propriétés-lien en `owl:negativeObjectPropertyAssertion` et pour les attributs en `owl:negativeDataPropertyAssertion`.

Nouveaux axiomes pour les propriétés

L'extension OWL 2 permet la définition d'ontologies plus expressives, car une transition est faite, entre la logique $\mathcal{SHOIN}$ qui est derrière OWL-DL, et la logique $\mathcal{SROIQ}$, par l'intermédiaire de l'ajout d'un ensemble de nouveaux axiomes [182]. Ceux-ci visent la définition de restrictions sur les propriétés, la spécification des caractéristiques des propriétés, la définition des chaînes de propriétés ou des clés. Concrètement, l'ensemble des nouveaux axiomes inclut :

- la restriction `ObjectHasSelf`, utilisée pour définir une classe comme étant composée de tous les objets qui sont reliés à eux-mêmes par une propriété-lien donnée. Exemple[13] :

```
SubClassOf(a:Narcissist ObjectHasSelf(a:likes))
```

[12] Structure syntaxique dédiée à faciliter l'expression d'axiomes complexes (tels que l'union d'un ensemble de classes disjointes), qui sont souvent rencontrés en pratique.

[13] La syntaxe utilisée pour les exemples qui suivent est la syntaxe du style fonctionnel de OWL 2 (en anglais *OWL 2 Functional-Style Syntax*).

- les restrictions de cardinalité qualifiées[14] et non-qualifiées, toutes deux supportées par OWL 2, autant pour limiter le nombre d'instances de propriétés-lien que pour limiter le nombre d'instances des attributs. Exemples :

```
ObjectExactCardinality(1 a:hasMother a:Woman)
ObjectMinCardinality(18 a:hasAge a:Adult)
ObjectMaxCardinality(5 a:hasDoor a:Car)
DataMinCardinality(1 a:hasPassportNumber)
```

- les caractéristiques des propriétés telles que la réflexivité, l'iréflexivité, la symétrie et l'antisymétrie définies pour des expressions à base de propriétés. Exemples :

```
ReflexiveObjectProperty(a:part_of)
IreflexiveObjectProperty(a:proper_part_of)
IrreflexiveObjectProperty(a:boundedBy)
AsymmetricObjectProperty(a:proper_part_of)
```

- les axiomes `DisjointObjectProperties` et `DisjointDataProperties` qui permettent de définir deux ou plusieurs expressions à base de propriétés comme étant disjointes deux à deux :

```
DisjointObjectProperties(a:touches a:disjoint a:contains)
DisjointDataProperties(a:startTime a:endTime)
```

- les chaînes de propriétés, qui permettent de définir les propriétés comme composition d'autres propriétés. En Logique de Description, ces axiomes sont connus sous le nom d'*inclusion de rôles complexes* [104]. À l'aide de chaînes de propriétés, on peut spécifier, par exemple, le fait que si un individu x est le propriétaire d'un objet y dont l'une des parties est z, alors x est également le propriétaire de z :

```
SubObjectPropertyOf(SubObjectPropertyChain(a:owns a:part) a:owns)
```

- l'axiome `HasKey`, qui permet de spécifier, pour une expression à base de classes, *CE*, un ensemble de propriétés dont les valeurs sont vues comme des clés qui identifient de façon unique les individus attachés à *CE*. En d'autres mots, cet axiome spécifie que tout individu nommé, instance de *CE* est identifié de façon unique par une propriété (lien ou attribut) ou un ensemble de propriétés. Alors, si pour deux individus nommés, les valeurs de chacune des propriétés clés coïncident, les deux individus sont identiques. Prenons comme exemple d'utilisation, la déclaration d'une clé pour les patients d'un hôpital, comme étant leur numéro d'enregistrement :

```
HasKey(a:Patient a:hasWaitingListNumber)
ClassAssertion(a:Patient a:Paul)
DataPropertyAssertion(a:hasRegistrationNumber a:Paul ''123-45-6879'')
```

Support pour les types définis par l'utilisateur

Le langage OWL 2 offre un ensemble étendu de types de base qui inclut plusieurs types de nombres (réels, décimaux, entiers positifs, *etc.*), les chaînes de caractères, les booléens, le type IRI[15], le type binaire, les types temporels, *etc.*, décrits en détails en [181]

Une caractéristique très importante de OWL 2 est le support qu'il offre pour la définition de types dérivés, construits en utilisant les facettes de restriction définies par le Schéma XML, appliquées aux types de base [182, 181]. Les facettes de restriction proposées par OWL 2 sont :

[14] L'attribut *qualifié* signifie que la restriction n'est pas définie pour le nombre total de valeurs de la propriété visée, mais pour le nombre de valeurs ayant un certain type (*i.e.* classe ou type de données)

[15] Le standard IRI (Identificateurs Internationalisés de Ressource) permet aux développeurs de contenu et aux utilisateurs d'écrire des identificateurs de ressources dans leur propre langue. Tout URI est un IRI.

– `xsd:minLength`, `xsd:maxLength`, `xsd:length`, `xsd:pattern` pour les chaînes de caractères (*i.e.* `xsd:string`, `xsd:normalizedString`, `xsd:token`, *etc.*) et pour les IRI ;
– `xsd:minInclusive`, `xsd:maxInclusive`, `xsd:minExclusive`, `xsd:maxExclusive` pour les types réels (`xsd:double` et `xsd:float`) ainsi que pour les types temporels (`xsd:dateTime` et `xsd:dateTimeStamp`) ;
– `xsd:minLength`, `xsd:maxLength`, `xsd:length` pour les types binaires (`xsd:hexBinary`, `xsd:base64Binary`) ;

Ces nouveaux types, tels que les types `a:SSN` et `a:adult` définis dans l'exemple 3.2, sont obtenus en imposant des restrictions sur les types de base, dans notre cas les types `xsd:string` et `xsd:integer`, et peuvent être utilisés dans des axiomes comme des types de données prédéfinis.

Exemple 3.2. *Exemple de types dérivés :*

```
DatatypeDefinition(
    a:SSN
    DatatypeRestriction( xsd:string xsd:pattern "[0-9]{3}-[0-9]{2}-[0-9]{4}" )
)
DatatypeDefinition(
    a:adult
    DatatypeRestriction( xsd:integer minInclusive "18"^^xsd:integer)
)
SubClassOf(a:Adult
           DataSomeValuesFrom(
              a:age
              a:adult         )
)
```

Expressivité croissante : de OWL 2 DL à OWL 2 Full

OWL 2 DL. À l'image de son prédécesseur, OWL DL, OWL 2 DL s'adresse aux utilisateurs qui souhaitent une expressivité maximale sans perdre la complétude et la décidabilité des raisonnements. Ainsi, OWL 2 DL impose plusieurs contraintes sur les déclarations et les axiomes acceptés, afin d'assurer une compatibilité stricte avec la logique $\mathcal{SROIQ}$[16], et, en conséquence, la décidabilité des raisonnements [181]. Il s'agit notamment d'une séparation stricte entre les ensembles de IRI qui identifient les *propriétés-lien*, les *attributs*, et les *propriétés d'annotation*, déclarées dans la fermeture transitive d'une ontologie[17]. La même séparation est réalisée au niveau des *classes* et des *types de données*. L'objectif de ces séparations est de pouvoir identifier de façon non ambiguë le type d'un IRI lors du traitement ou de la manipulation d'une ontologie.

De plus, les IRI utilisés pour identifier les entités ne peuvent pas appartenir au vocabulaire réservé de OWL 2, à quelques exceptions près : `owl:Thing`, `owl:Nothing`, `owl:topObjectProperty`, `owl:bottomObjectProperty`, `owl:topDataProperty`, `owl:bottomDataProperty` et les IRI identifiant les types de données du *datatype map* de OWL 2.

À ces restrictions s'ajoute un ensemble de contraintes appliquées aux axiomes contenus par une ontologie OWL 2. Ainsi, si on considère *Ax* comme étant l'ensemble des axiomes définis dans

[16] SROIQ = SHOIQ + axiomes qui permettent de définir des rôles disjoints, réflexifs, irréflexifs, des inclusions complexes de rôles, la réflexivité locale, et de déclarer des négations de rôles.

[17] La fermeture transitive d'une ontologie *O* inclut l'ontologie elle-même, toutes les ontologies importées par *O*, et de façon récursive toutes les ontologies importées par les ontologies importées par *O*.

la fermeture transitive d'une ontologie OWL 2 DL, *O*, alors *Ax* doit satisfaire toutes les contraintes suivantes :

a) la propriété `owl:topDataProperty` peut être spécialisée par les autres propriétés, à travers des axiomes `SubDataProperty`, mais elle ne spécialise aucune autre propriété ;

b) tout type de données utilisé dans un axiome de *Ax* est, soit défini par le *datatype map*, soit introduit par un axiome `DatatypeDefinition` de *Ax*. De plus, une relation d'ordre partiel stricte existe entre les types de données, dont l'objectif est d'éliminer les définitions de types de données cycliques.

c) les expressions à base de classes définies en utilisant les restrictions `ObjectMinCardinality`, `ObjectMaxCardinality`, `ObjectExactCardinality`, et `ObjectHasSelf`, ainsi que les axiomes `FunctionalObjectProperty`, `InverseFunctionalObjectProperty`, `AsymmetricObjectProperty`, `IrreflexiveObjectProperty` et `DisjointObjectProperties` peuvent faire référence exclusivement à des *propriétés-lien simples* (telles que définies dans [181]) :

d) un ordre partiel strict est imposé pour les propriétés-lien définies en utilisant des chaînes de propriétés, afin d'éliminer les définitions cycliques sur des chaînes de propriétés, pour lesquelles les raisonnements ne sont pas décidables ;

e) les axiomes `NegativeObjectPropertyAssertion`, `NegativeDataPropertyAssertion`, `DifferentIndividuals` et `SameIndividual` ne peuvent pas contenir des individus anonymes ;

f) une ontologie OWL 2 DL qui contient des individus anonymes, peut être transformée en une ontologie équivalente contenant exclusivement des individus nommés.

Pour que l'ontologie *O* soit considérée conforme à OWL 2 DL, toute ontologie O' importée par *O* doit également satisfaire les conditions présentées auparavant.

OWL 2 Full. Les ontologies qui ne respectent pas toutes les contraintes évoquées auparavant sont considérées comme des ontologies OWL 2 Full.

Sémantique formelle

Le langage OWL 2 met a disposition deux types d'interprétation formelle pour les ontologies.

La sémantique directe. Réglementée dans [180], la sémantique directe propose, comme l'évoque son nom, une interprétation directe des structures ontologiques contrairement à la sémantique basée sur RDF, qui réalise l'interprétation d'une ontologie en la transformant d'abord en un graphe RDF. Elle est compatible avec la sémantique définie par la théorie des modèles pour la Logique de Description $\mathcal{SROIQ}$ - un fragment de la logique du premier ordre pour lequel des algorithmes de raisonnement décidables et efficaces ont été définis [5]. Grâce à ce rapprochement, les différents outils construits pour les ontologies OWL 2 peuvent exploiter les travaux théoriques réalisés autour des Logiques de Description, ainsi que l'expérience d'implémentation obtenue lors de la construction de divers raisonneurs [5]. Seuls les ontologies OWL 2 DL peuvent être interprétées en utilisant la *sémantique directe* [5].

La sémantique basée sur RDF. La sémantique basée sur RDF [214] propose une interprétation indirecte des ontologies OWL 2, à travers un *mapping* entre OWL 2 et les graphes RDF [5]. Elle est compatible avec la sémantique attachée aux graphes RDF [127] qu'elle étend. La sémantique basée sur RDF peut être utilisée pour interpréter toute ontologie OWL 2, sans restriction, car toute ontologie OWL 2 peut être transformée en un graphe RDF [5]. L'appellation *OWL 2 Full* est utilisée de façon informelle pour désigner les graphes RDF considérés comme des ontologies OWL 2 et interprétés avec la sémantique basée sur RDF [5].

3.2.3 Outils de modélisation

De nombreux outils pour la construction des ontologies ont été développés ces dernières années. Nous présentons dans ce qui suit les outils les plus largement utilisés par la communauté ingénierie des ontologies.

3.2.3.1 OILEd

OILEd[18] est un éditeur graphique simple, qui permet à l'utilisateur de construire des ontologies en langage OIL (Ontology Interchange Language), une extension de RDF(S) qui inclut un ensemble étendu de primitives de modélisation, et qui est équivalent à la logique $\mathcal{SHIQ}$ [28]. La construction de OILEd s'est concentrée sur l'étude des évolutions du paradigme de *frames* pour qu'il puisse mieux gérer les langages de modélisation plus expressifs et sur l'utilisation des raisonneurs comme support pour la création et la gestion d'ontologies [28].

Développé par l'Université de Manchester, OILEd n'a officiellement pas d'autre ambition que de construire des exemples montrant les vertus du langage pour lequel il a été créé. À ce titre, OILEd est souvent considéré comme une simple interface de la logique de description $\mathcal{SHIQ}$ [55]. Néanmoins, il offre la plus grande partie des fonctionnalités que l'on peut attendre d'un éditeur d'ontologies. On peut créer des hiérarchies de classes et spécialiser les rôles, et utiliser avec l'interface les types d'axiomes les plus courants [55]. Cet éditeur offre également les services du raisonneur FaCT[19] (Fast Classification of Terminologies), qui permet de tester la satisfaisabilité des définitions de classes et de découvrir des subsomptions implicites de l'ontologie.

3.2.3.2 KAON et KAON2

KAON[20] est une plate-forme *open-source* développée par l'Université de Karlsruhe, dédiée à la création, la manipulation et la gestion d'ontologies et spécialisée dans des applications d'entreprise. Elle est centrée sur l'intégration de technologies de gestion d'ontologies avec celles propres au domaine de l'entreprise, et notamment les bases de données relationnelles [24]. KAON est utilisé dans des projets de recherche qui s'intéressent particulièrement à la réutilisation et à l'évolution des ontologies [24].

KAON2 est le successeur du projet KAON (souvent nommé KAON1). La différence principale entre les deux vient du langage d'ontologie utilisé comme support : KAON1 utilise une extension propriétaire de RDF(S), alors que KAON2 est basé sur OWL-DL. Parmi les fonctionnalités mises à disposition par KAON2 on peut citer :

[18] http ://oiled.man.ac.uk/
[19] http ://www.cs.man.ac.uk/ horrocks/FaCT/
[20] http ://kaon.semanticweb.org/

– une API pour la gestion d'ontologies OWL-DL, SWRL [134], et F-Logic [156] ;
– un serveur autonome qui permet l'accès aux ontologies dans une manière distante, à travers RMI[21] ;
– un moteur d'inférence capable de répondre à des requêtes conjonctives (exprimées en utilisant la syntaxe SPARQL [204]) ;
– une interface DIG[22], permettant l'accès à partir d'outils tels que Protégé (voir section 3.2.3.4) ;
– un module d'extraction d'ontologies à partir de bases de données relationnelles.

3.2.3.3 Jena

Jena [163] est une plate-forme open source, développée dans le cadre du projet *HP Labs Semantic Web Program* avec l'objectif de contribuer à la construction d'applications pour le Web Sémantique. Jena fournit un environnement de programmation Java pour la création et la manipulation de graphes RDF, RDFS et d'ontologies OWL. Il inclut un moteur d'inférence à base de règles. La plate-forme est construite autour d'une abstraction simple des graphes RDF [238], utilisée de façon uniforme pour gérer les graphes complètement chargés en *mémoire* ou en couplage avec des bases de données, et les graphes inférés.

La contribution principale de Jena réside dans son API pour la manipulation des graphes RDF [238], mais la plate-forme inclut également des outils divers comme, par exemple, un analyseur RDF/XML, un langage d'interrogation, des modules de lecture/écriture pour N3[23], etc.

3.2.3.4 Protégé

Protégé est une plate-forme de développement d'ontologies et d'acquisition de connaissances développée par *Stanford Medical Informatics, Standford University School of Medicine*. Logiciel *open-source*, Protégé est muni d'une interface graphique qui permet aux développeurs d'ontologies de se concentrer sur leur tâche de modélisation conceptuelle, sans connaître, ni penser à la syntaxe d'un langage de représentation.

Le modèle de connaissances de Protégé est flexible et son architecture est facilement extensible à travers des *plugins*. Ces deux caractéristiques rendent l'outil rapidement adaptable à de nouveaux langages du Web Sémantique [24]. À ce jour il dispose de *plugins* pour les langages CLIPS, XML, UML, RDF, DAML+OIL, OWL, OWL2, *etc.* qui permettent d'utiliser Protégé comme éditeur d'ontologies pour ces différents langages, de les peupler avec des instances et de les sauver dans les différents formats.

La plate-forme Protégé offre deux possibilités de modélisation d'ontologies :

a) en utilisant l'éditeur *Protégé-Frames*, qui permet à l'utilisateur de créer et peupler des ontologies définies selon le modèle de *frames*, et conformes au protocole Open Knowledge Base Connectivity[24] (OKBC). Dans ce modèle, une ontologie se compose de :

[21] *Remote Method Invocation*, plus connu sous l'acronyme RMI, est une interface de programmation (API) pour le langage Java qui permet d'appeler des méthodes distantes.

[22] L'interface DIG est une interface XML standardisée pour les systèmes basés sur les Logiques de Description, définie par *DL Implementation Group*.

[23] Notation3 ou N3 est une sérialisation alternative pour les modèles RDF, conçue avec l'objectif d'offrir une meilleure lisibilité pour l'utilisateur : N3 est beaucoup plus compacte et plus lisible que la notation XML RDF.

[24] http ://www.ai.sri.com/ okbc/

- un ensemble de classes, organisées dans une hiérarchie de subsomption, qui représentent les principaux concepts du domaine modélisé,
- un ensemble de *slots* associés aux classes et qui décrivent leurs attributs et leurs propriétés-lien,
- un ensemble d'instances de classe.

b) à travers l'éditeur *Protégé-OWL*. Comme son nom l'indique, Protégé-OWL est une extension de l'éditeur d'ontologie Protégé pour manipuler le format OWL. Il permet notamment de visualiser, d'éditer classes et propriétés OWL, de communiquer avec des raisonneurs logiques ou de peupler l'ontologie d'instances trouvées dans des documents. Visuellement, le logiciel se présente sous la forme de plusieurs onglets permettant l'accès à différents types d'information : on peut citer, entre autres, les onglets *Classes*, *Properties* (avec une subdivision supplémentaire pour chaque type de propriété OWL) et *Individuals*.

Protégé est également une librairie Java qui peut être étendue pour créer de véritables applications à base de connaissances en utilisant un moteur d'inférence pour raisonner et déduire de nouveaux faits par application de règles d'inférence aux instances de l'ontologie (ABox) et à l'ontologie elle-même (TBox) [54]. Il est aussi possible de raisonner sur les ontologies en utilisant un moteur d'inférence général tel que JESS[25], ou des outils d'inférence spécifiques au Web Sémantique basés sur des Logiques de Description tels que RacerPro ou Fact++ (voir section 3.2.4). Ces deux outils peuvent être facilement intégrés à Protégé.

Une nouvelle version de l'éditeur, Protégé 4.0, a été récemment développée dans le cadre du projet CO-ODE[26], au-dessus de l'API open source de OWL[27]. Celle-ci permet la création et la manipulation d'ontologies OWL 2 et fournit un ensemble étendu de nouvelles fonctionnalités, comme :

- le chargement de plusieurs ontologies dans le même espace de travail ;
- la fusion d'ontologies et la suppression d'importations redondantes ;
- le déplacement des axiomes d'une ontologie vers une autre ;
- l'utilisation intelligente d'entrepôts locaux/globaux pour une meilleure gestion des dépendances,
- le passage dynamique entre les différents ontologies actives ;
- la possibilité de configurer de façon persistante la composition et la mise en page de l'éditeur ;
- la création, l'importation et l'exportation des onglets configurés par l'utilisateur ;
- plusieurs vues alternatives sur la même ontologie ;
- *etc.*

Au niveau des raisonnements, le nouvel outil dispose d'un onglet dédié (DL Query) pour tester des expressions de classe arbitraires, et possède des connexions prédéfinies avec les raisonneurs FaCT++ et Pellet.

3.2.4 Outils d'inférence

Actuellement, il existe plusieurs moteurs d'inférence, la plupart conçus pour raisonner sur les Logiques de Description, mais qui acceptent en entrée des fichiers OWL/RDF(S). Parmi ceux-ci,

[25] http ://herzberg.ca.sandia.gov/jess/
[26] http ://www.co-ode.org/
[27] http ://owlapi.sourceforge.net/

on peut citer RacerPro [205], Pellet [58], Fact [133], Fact++ [230], Surnia [126], F-OWL [57], Hoolet [6], *etc.* Certains moteurs d'inférence ne peuvent raisonner qu'au niveau terminologique, alors que des moteurs comme Pellet et RacerPro permettent de raisonner aussi sur les instances de concepts. Nous les présentons ci-dessous.

3.2.4.1 RacerPro

Racer [205], acronyme de "Renamed Abox and Concept Expression Reasoner", est un système de représentation des connaissances qui implémente un algorithme de type "tableau calculus" optimisé, pour la Logique de Description $\mathcal{SHIQ}$. Il peut traiter des documents OWL Lite et DL, mais ignore les types définis par l'utilisateur et les classes énumérées [205]. Il se présente sous la forme d'un serveur qui peut être accédé par le protocole TCP ou HTTP. Comme services fournis par RacerPro on peut énumérer [205] :

- la vérification de la consistance d'une ontologie[28] ;
- l'inférence de relations de subsomption implicites, déduites à partir des axiomes et des faits déclarés explicitement dans une ontologie ;
- la découverte des synonymes d'une ressource (classe ou instance) ;
- la recherche d'informations contenues dans les documents OWL. À ces fins, RacerPro propose un système de traitement des requêtes basé sur le langage *OWL-QL Query* [92] ;
- l'importation de ressources depuis d'autres ontologies à l'aide d'un client HTTP ;
- le traitement incrémental de requêtes, qui consiste à trouver, pour une requête donnée, les *n* réponses les plus immédiates dans un but de minimiser les calculs. Les réponses qui impliquent le moins de calculs sont livrées les premières, et l'utilisateur a la latitude de décider s'il faut continuer à calculer toutes les réponses possibles.

De plus, RacerPro peut réaliser des raisonnement algébriques comme :

- traiter des restrictions min/max sur des entiers ;
- résoudre des équations polynomiales impliquant des réels ou des cardinaux avec des relations d'ordre ;
- vérifier l'égalité ou l'inégalité des chaînes des caractères.

Une description complète du raisonneur peut être consultée dans [205].

3.2.4.2 Pellet

Pellet [221, 58] est une librairie *open-source* Java qui implémente des raisonnements applicables aux ontologies OWL. Elle est utilisable en conjonction avec Jena et les API OWL. Parmi les fonctionnalités offertes par Pellet, on peut citer :

- la vérification de la consistance d'une ontologie ;
- la classification pour les concepts des taxonomies ;
- les tests de subsomption ;
- l'analyse et réparation des ontologies qui implique une vérification des documents OWL dans la syntaxe RDF, avec le but d'identifier et de corriger les fautes de syntaxe. Cela impose qu'un document soit écrit en OWL Full ;
- le raisonnement sur des types de données.

[28] Une ontologie est *consistante* si et seulement si elle est *satisfiable* dans le sens où elle est sémantiquement correcte et ne présente pas de contradiction logique.

Pellet est un raisonneur puissant basé sur l'algorithme des tableaux, le seul qui prenne en charge pour les classes énumérées et les types de données définis par l'utilisateur [166, 221]. Pour résoudre le problème des URI qui désignent les nouveaux types, Pellet adopte la solution proposée par DAML+OIL : la construction d'un nouvel URI à partir de l'URI du document contenant le schéma XML et le nom local du type simple [166]. Le document identifié par l'URI et qui contient le schéma XML sera analysé avec le but de localiser le type dont l'attribut `name` contient le nom de la définition. En conséquence, OWL pourra utiliser des types de données simples définis au premier niveau d'un document XML Schéma, en les référençant par l'URI du document et le nom local du type [166].

La nouvelle version, Pellet 2.0 est compatible avec le langage OWL 2 [58] et intègre diverses techniques d'optimisation, y compris pour les nominaux, les requêtes conjonctives, le raisonnement progressif, *etc.* [58].

Les recommandations à l'initiative du Web Sémantique, suffisamment génériques, permettent de répondre avec succès à la plupart des exigences d'expressivité imposées par les divers domaines d'application abordés. Néanmoins, en termes de traitement de l'information spatio-temporelle, on observe une absence de modélisation standardisée [8] et de mécanismes de raisonnement dédiés [88, 190].

3.3 Représentation ontologique de l'espace pour le Web Sémantique Géospatial

Afin de combler les lacunes évoquées ci-dessus, un groupe de travail a été crée par le W3C, appelé Geospatial Incubator Group[29], dont le but est, dans un premier temps, d'étudier différentes modalités pour spécifier la localisation et les propriétés géographiques des ressources Web, et pour proposer, ensuite, des recommandations d'ontologies géospatiales standard.

Un premier résultat issu des travaux du groupe [153], propose une classification des ontologies spatiales en sept catégories :

a) **Ontologies géospatiales pour la description des *éléments* (en anglais *features*).** Ce type d'ontologie est destiné à fournir une représentation formelle des normes ISO et OGC pour la définition des *éléments* dans le contexte du Web Sémantique. Le vocabulaire géospatial défini par une ontologie de ce type représente un ensemble restreint du modèle général de *features* (GFM) (voir section 2.3.1), capable de représenter une proportion importante des ressources du Web. Cependant, un grand nombre d'applications telles que le routage, l'observation de la terre, et l'interprétation des images requièrent des représentations plus complètes. Plusieurs projets de traduction des standards ISO et OGC vers des ontologies OWL sont recensés à l'adresse : http ://loki.cae.drexel.edu/~wbs/ontologie/list.htm.

b) **Ontologies pour la définition de types d'*éléments***. Ces ontologies peuvent être considérées comme des collections de définitions d'éléments généralement reconnues dans une communauté étendue (ou même dans le monde). Un exemple de ce type d'ontologie, décrivant les limites administratives du Royaume-Uni est *AdministrativeGeography*[30], publiée

[29] http ://www.w3.org/2005/Incubator/geo/

[30] http ://www.ordnancesurvey.co.uk/oswebsite/ontology/v1/AdministrativeGeography.owl.

par l'*Ordnance Survey*[31].

c) **Ontologies pour la définition de relations spatiales**. Comme nous l'avons montré dans le chapitre 2, il existe un ensemble large de relations spatiales qualitatives qui ont été étudiées de façon intensive dans les dernières décennies. Également, le vocabulaire courant présente de nombreuses versions *plus familières* de ces relations spatiales, tels que "à coté de", "derrière", "en vis-à-vis de", "au-dessous de", *etc.* Une des nombreuses ontologies qui définissent un ensemble de relations spatiales est l'ontologie *SpatialRelations*[32], créée par l'*Ordnance Survey*.

d) **Ontologies de noms de lieux (toponymes)**. Les noms de lieux sont un moyen très utilisé pour la géolocalisation des ressources, avec, évidement, un certain degré d'approximation. Plusieurs travaux ont porté sur la définition des ontologies et des taxonomies de lieux, mais ils n'ont pas vraiment été réunis pour constituer une référence commune [153]. Une référence utile est l'ontologie *GeoNames*[33].

e) **Ontologies définissant des systèmes de coordonnées.** L'OGC et l'ISO fournissent des normes (par exemple, ISO 19111) qui définissent des modèles et des implémentations pour les systèmes de coordonnées de référence. Plusieurs projets d'ontologies qui traduisent ces normes vers OWL ont vu le jour, notamment l'ontologie définie par l'Université Drexel, Iso-19111[34].

f) **Meta-ontologies géospatiales.** La qualité des données, ainsi que leur provenance, sont essentielles pour une exploitation efficace des données géographiques et de géolocalisation, sur le Web. Une approche générale pour la description des méta-données est définie par la norme ISO 19115, et les règles de codage XML sont spécifiées dans l'ISO 19139. Des questions fondamentales telles que la pertinence et la validité temporelle d'une ressource géospatiale sur le Web (par exemple, l'image d'une carte), ainsi que la représentation de ces propriétés par des URI ne sont pas encore résolues [153].

g) **Ontologies géospatiales pour les services Web.** L'évolution des normes telles que OWL-S[35] pose des défis en termes de mise en place de descriptions formelles et actionnables pour les services Web. Selon [153], des précisions sont encore nécessaires quant à la manière dont le contenu de la plupart des services Web géospatiaux interagit avec leur modèle de processus et avec le comportement attendu des interfaces. Plusieurs travaux sur la sémantique géospatiale de modèles de services sont étudiés dans [151].

Dans ce travail, nous nous concentrons principalement sur les ontologies géospatiales pour la description des *éléments* ou *features* (a), qui définissent des concepts et des propriétés dédiés à la description des informations spatiales quantitatives (voir section 3.3.1), et sur les ontologies pour la définition de relations spatiales (c), qui sont utiles pour la description des données spatiales qualitatives (voir section 3.3.2).

[31] *Ordnance Survey* est l'agence nationale britannique chargée de la cartographie.
[32] http ://www.ordnancesurvey.co.uk/ontology/SpatialRelations.owl
[33] http ://www.geonames.org/ontology/
[34] http ://loki.cae.drexel.edu/~wbs/ontology/2004/09/iso-19111.owl
[35] OWL-S est une ontologie OWL qui fournit le vocabulaire de base pour la description des propriétés et des fonctionnalités des services Web.

3.3.1 Modélisation spatiale quantitative en OWL

On constate que le langage OWL ne propose pas de types spatiaux. La raison à cela est que les raisonnements à l'aide de types complexes n'est pas décidable [158] (voir la section 3.2.2.3). Néanmoins, comme il est rappelé dans la section 3.3, il existe de nombreux travaux qui proposent des ontologies pour la description d'*éléments* (ou *features*), dans lesquelles des concepts spatiaux remplacent/simulent les types spatiaux. Parmi celles-ci, nous présentons dans cette section, l'ontologie GeoRSS, devenue recommandation du consortium W3C comme vocabulaire de référence pour la description des propriétés géospatiales des ressources Web [152].

D'une manière générale, l'activité du groupe de travail *Geospatial Incubator Group* (GeoXG) a été largement influencée par les travaux du consortium OGC, de l'ISO et de l'organisation GeoRSS. Si la rigueur des modèles OGC et ISO est indispensable pour la clarté des représentations spatiales, leur profondeur et leur complexité vont au-delà des besoins de la plupart des applications Web [153]. En conséquence, le groupe a essayé d'adapter, pour le Web, les modèles spatiaux standard existants, notament le modèle spatial simplifié, appelé *GeoRSS*. Ce dernier est illustré dans la figure 3.5.

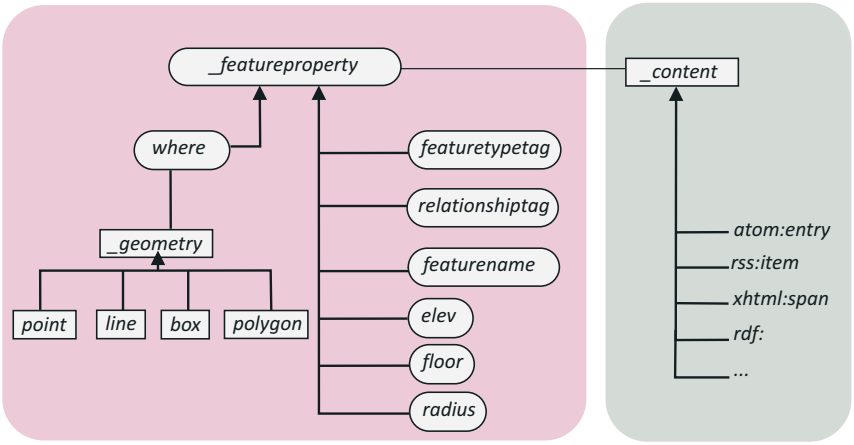


FIG. *3.5: Représentation graphique du modèle GeoRSS.*

Au centre du modèle GeoRSS, se trouve la relation générique, `_featureproperty`, qui peut être utilisée pour définir l'emprise spatiale d'une ressource Web (`_content`) à travers la spécification d'une géométrie. La relation `where`, spécialisation de `_featureproperty`, est utilisée pour associer l'entité spatiale (`_content`) à un type de géométrie (`_geometry`), choisi parmi un ensemble limité de concepts qui représentent les types spatiaux de base : `point`, `line`, `box` et `polygon`. Ces derniers permettent d'associer une représentation numérique quantitative aux éléments (*features*) et facilitent l'analyse et la visualisation des données. D'autres spécialisations de `_featureproperty` décrivent des attributs utilisés habituellement pour caractériser les *features*, tels que le nom (`featurename`), le type (`featuretype`), l'élévation (`elev`), *etc.*

Le modèle GeoRSS dispose de plusieurs encodages : Atom, GML, XML, RDF/RSS ainsi que OWL. L'encodage OWL du modèle, *GeoRSS-Simple*[36], est un format très léger pour la description de données spatiales que les développeurs et les utilisateurs peuvent rapidement et facilement intégrer dans leurs applications, ou bien utiliser pour créer des annotations sémantiques avec peu

[36] http ://mapbureau.com/neogeo/neogeo.owl

d'efforts. Au centre de l'ontologie, on trouve la classe `gml:_Feature`[37] qui regroupe tous les types d'éléments spatiaux, et peut être reliée à une description géométrique (`gml:_Geometry`) à travers l'utilisation de la propriété-lien `where`. L'ontologie définit les géométries de base - `gml:Point`, `gml:Polygon`, `gml:LigneString`, `gml:LinearRing`, `gml:Envelope` - comme spécialisations de la classe abstraite `gml:_Geometry` (voir figure 3.6).

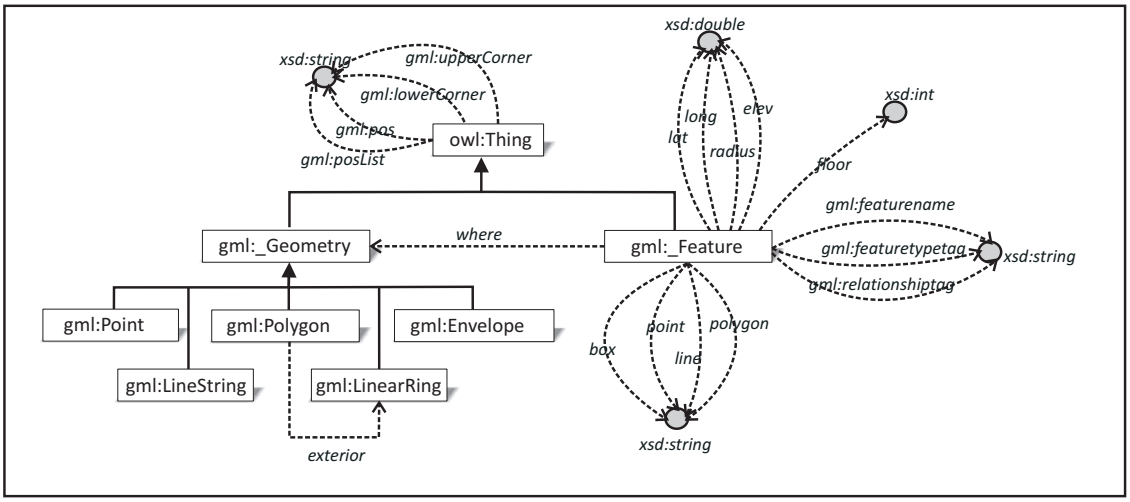


FIG. *3.6: Représentation graphique de l'ontologie GeoRSS Simple.*

Plusieurs attributs - `box`, `point`, `line` et `polygon` - peuvent être utilisées pour attacher aux objets `gml:_Feature`, des géométries concrètes, spécifiées en utilisant des chaînes de caractères qui respectent un format donné. Par exemple, le fragment OWL suivant définit les coordonnées géographiques de la tour Eiffel.

```
<_Feature rdf:about="tourEiffel">
      <georss:point rdf:datatype="&xsd;string">
           48.8583 2.2945
      </georss:point>
</_Feature>
```

Pour définir une ligne (`line`), il faut fournir au moins deux paires de coordonnées (latitude et longitude), séparées par des espaces. Les polygones nécessitent au moins quatre paires latitude, longitude, la première paire devant être égale à la dernière. L'exemple suivant montre une description possible du bâtiment du Pentagone.

```
<_Feature  rdf:about="Pentagon">
      <featurename rdf:datatype="&xsd;string"> Pentagon </featurename>
      <georss:where rdf:resource="pol1"/>
</_Feature >
<gml:Polygon rdf:about="pol1">
      <gml:exterior rdf:resource="linRing1"/>
```

[37] Le préfixe gml vient de *Geography Markup Language* (GML), un langage de type XML pour encoder, manipuler et échanger des données géographiques. Ce dernier est un standard développé par l'Open Geospatial Consortium pour garantir l'interopérabilité des données dans le domaine de l'information géographique et de la géomatique. GML consiste en un ensemble de schémas XML qui définissent un format ouvert pour l'échange de données géographiques, et permettent de construire des modèles de données spécifiques pour des domaines spécialisés, comme l'urbanisme, l'hydrologie ou la géologie.

```
</gml:Polygon>
<gml:LinearRing rdf:about="lr1">
      <gml:posList rdf:datatype="&xsd;string">
          -77.0579537082376 38.87258672915797
          -77.0584754972063 38.87005708744568
          -77.0555999760046 38.86886750371786
          -77.0532658178173 38.87064560343153
          -77.0546559466248 38.87292421787603
          -77.0579537082376 38.87258672915797
      </gml:posList>
</gml:LinearRing>
```

Il est également possible de caractériser un élément spatial à travers l'approximation de sa localisation par le rectangle englobant minimal. Celui-ci peut être spécifié par deux points de l'espace géographique, désignés par les attributs `gml:lowerCorner` et `gml:upperCorner` dont les valeurs sont des chaînes de caractères. Toujours pour approximer une région, l'attribut `radius`, indique la taille en mètres d'un rayon autour de la géométrie d'un objet. L'attribut `elev` permet de spécifier l'élévation d'un point, qui est une donnée aisée à obtenir en utilisant un dispositif GPS usuel. Pour décrire le nombre d'étages d'un bâtiment, on dispose de l'attribut `floor` dont la valeur est un entier. Les caractéristiques des objets `gml:_Feature` sont modélisés à travers des attributs dont les valeurs sont des chaînes de caractères, tels que : `gml:featurename`, `gml:featuretypetag` et `gml:relationshiptag`.

3.3.2 Modélisation spatiale qualitative en OWL

Alors que le *GeoXG* a achevé les travaux de spécification du modèle géométrique de représentation en proposant une ontologie OWL qui traduit ce modèle pour le Web Sémantique, la situation est différente en ce qui concerne la représentation des relations spatiales qualitatives. En soulignant l'importance et l'utilité de celles-ci, le groupe de travail cite l'ensemble des relations RCC8 comme étant utilisées le plus souvent pour décrire des configurations spatiales. Bien que ces relations spatiales soient gérées de façon adaptée et efficace dans le domaine des sciences géographiques, notamment par les SIG, leur formalisation et leur adaptation pour le Web Sémantique est encore dans une phase peu avancée. Ainsi, la spécification formelle des relations spatiales à travers l'utilisation du langage OWL est-elle citée comme travail futur du groupe *GeoXG* [152, 151].

En l'absence d'une recommandation du W3C pour un vocabulaire de relations spatiales qualitatives, nous avons étudié le travail de l'*Ordnance Survey*, qui propose l'ontologie SpatialRelations[38]. Celle-ci est structurée autour de deux concepts : celui d'emprise spatiale (`Footprint`), et celui de région spatiale (`SpatialRegion`). Les objets de type `Footprint` permettent de relier n'importe quel type d'objet spatial à sa représentation géométrique, à travers l'utilisation de la propriété-lien `hasFootprint`.

Une hiérarchie de relations topologiques, inspirée de l'algèbre RCC8, est définie comme reliant des régions spatiales, instances de la classe `SpatialRegion` (voir figure 3.7). La racine de cette hiérarchie, `spatiallyRelated`, dispose de deux spécialisations `spatiallyConnected` et `spatiallyDisjoint` qui permettent de spécifier deux régions comme étant connectées, ou respectivement disjointes. Deux types de connexion générique entre régions sont pris en compte : l'inclu-

[38] http ://www.ordnancesurvey.co.uk/oswebsite/ontology/SpatialRelations.owl

sion spatiale (g_spatiallyContains[39] et son inverse g_spatiallyInside) et le chevauchement (g_spatiallyOverlapping). En termes d'algèbre RCC8, la propriété-lien g_spatiallyContains, est équivalente à la disjonction des relations *partie tangentielle propre* (TPP), modélisée par la classe s_spatiallyContains, *partie tangentielle non-propre* (NTPP), modélisée par la classe t_spatiallyContains, et *égalité de régions* (EQ), modélisée par la classe spatiallyEqual. Les propriétés-lien s_spatiallyInside et t_spatiallyInside traduisent les relations TPPi, respectivement NTPPi. La définition des relations TPP, NTPP, TPPi, NTPPi et EQ est donnée dans le chapitre 2.

Deux types de relation de chevauchement sont définies : la connexion (spatiallyTouching) et l'intersection (s_spatiallyOverlapping), toutes deux sont symétriques. Les relations spatiallyConnected, g_spatiallyOverlapping et spatiallyDisjoint sont également symétriques, tandis que les relations s_spatiallyContains, g_spatiallyInside, s_spatiallyInside sont transitives.

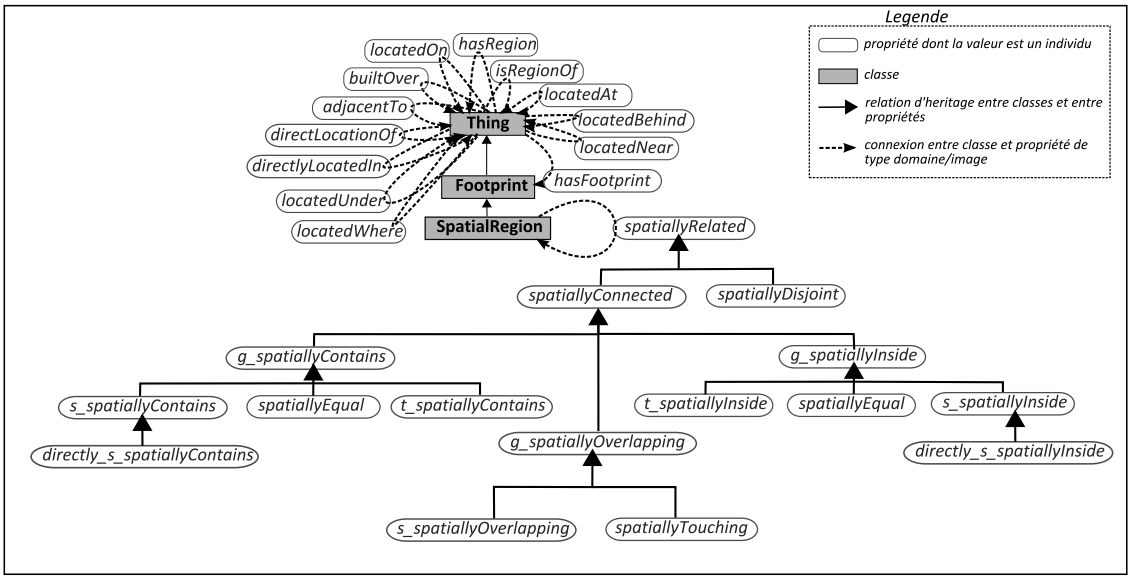


FIG. *3.7: Représentation graphique de l'ontologie SpatialRelations définie par Ordnance Survey.*

Outre les relations spatiales topologiques, l'ontologie *SpatialRelations* définit, pour tout type d'objet, deux relations méréologiques : hasRegion et son inverse isRegionOf, ainsi que quatre relations de direction locatedBehind, locatedOn, locatedUnder et builtOver. Pour modéliser la proximité, on peut faire appel à la propriété locatedNear.

3.3.3 Raisonnement spatial pour OWL

Les travaux que nous avons étudiés dans cette thèse ne proposent pas d'inférences basées sur des calculs géométriques, capables d'exploiter les connaissances quantitatives décrites dans des ontologies OWL.

Concernant la représentation spatiale qualitative, les raisonnements en OWL sont pour l'instant limités à l'exploitation de caractéristiques telles que l'existence d'un *inverse*, la *transitivité*, la

[39] Le préfixe g dans le nom d'une relation vient de "general", le préfixe s vient de "strict" et le préfixe t vient de "tangent".

symétrie, etc., des relations spatiales. Ainsi, les raisonneurs que nous avons étudiés dans la section 3.2.4 (Pellet, RacerPro) ne font pas de distinction entre l'information thématique et l'information spatiale. L'absence actuelle de raisonneurs spatiaux pour les ontologies OWL a deux causes principales. D'une part, il faut tenir compte du fait que la déduction des relations topologiques, de direction et de distance se base principalement sur des tableaux de composition (voir chapitre 2) qui ne peuvent pas être facilement exprimés et exploités à travers le langage OWL. D'autre part, les raisonnements logiques, tels que la vérification de la cohérence d'une ontologie, à base de concepts spatiaux sont en général NP complets.

Une approche intéressante de raisonnement à base de relations topologiques, dans une version étendue de la logique $\mathcal{ALC}(\mathcal{D})$, est proposée dans [122]. Nous la présentons très brièvement ci-dessous.

Extension spatiale des Logiques de Description.

Haarslev *et al.* ont proposé une extension de la logique de description $\mathcal{ALC}(\mathcal{D})$ appelée $\mathcal{ALCRP}(\mathcal{D})$ [122] pour raisonner sur les relations qualitatives entre les régions spatiales, et sur les propriétés des données quantitatives. L'extension proposée gère les objets spatiaux et leurs relations à l'aide de prédicats ($\mathcal{RP}$ signifie Rôles basés sur des Prédicats). Un domaine concret `Polygone` est introduit pour gérer la géométrie des objets.

Les relations topologiques entre les objets sont décrites à l'aide de huit prédicats élémentaires : `equal` (égalité), `disjoint` (disjonction), `touching` (adjacence), `s_overlapping` (recouvrement strict), `t_inside` (inclusion tangentielle), `s_inside` (inclusion stricte) et les relations inverses de ces deux dernières relations. Une définition formelle de ces relations est donnée dans [122] qui introduit également une relation composite comme disjonction de relations élémentaires.

De nouveaux concepts peuvent être décrits en utilisant l'opérateur de construction de concepts $\exists f.P$. On peut, par exemple, définir le concept de `PlageRoumaine` comme étant un espace adjacent à la `MerNoire` et à l'intérieur de la `Roumanie` :

$$PlageRoumaine = \exists \texttt{has\_Area}.\texttt{touching}(MerNoire) \wedge \exists \texttt{has\_Area}.\texttt{s\_inside}(Roumanie).$$

On suppose ici que tout objet possède un attribut `has_Area` qui décrit sa représentation spatiale polygonale.

La proposition de Haarslev est dans la lignée d'autres travaux proposant des extensions de Logiques de Description à l'aide de domaines concrets [226]. La capacité de définition des rôles topologiques facilite la spécification des objets spatiaux. Le domaine concret considéré donne accès à des algorithmes de raisonnement spatial permettant d'étendre les raisonnements terminologiques vers la dimension spatiale.

Toutefois, pour satisfaire les besoins d'applications spatiales complexes, comme les guides touristiques qui exploitent des coordonnées GPS correspondant à des *points* ou des routes modélisées comme des *lignes*, *etc.*, des extensions supplémentaires des Logiques de Description sont nécessaires, afin que ces dernières puissent être utilisées comme base de raisonnement pour le Web Sémantique Géospatial. De plus, ce type d'approche logique ne permet pas la manipulation de données géométriques quantitatives.

3.4 Représentation ontologique du temps pour le Web Sémantique Géospatial

Alors que le langage OWL n'intègre pas de types spatiaux, et qu'il n'existe pas à ce jour d'ontologie standard pour la représentation des concepts et relations spatiaux, les choses sont différentes en ce qui concerne l'information temporelle. Dans les sections qui suivent nous présentons les types temporels mis à disposition par OWL et complétés par l'ontologie OWL-Time, recommandation du W3C pour la modélisation des concepts temporels. Nous illustrons, ensuite, les principaux raisonnements temporels activables sur des données temporelles décrites en OWL.

3.4.1 Modélisation temporelle quantitative en OWL

Afin de modéliser des données temporelles en OWL, on peut exploiter les types dédiés mis à disposition, soit : `xsd:dateTime`, `xsd:date`, `xsd:gYearMonth`, `xsd:gMonthDay`, `xsd:gDay`, `xsd:gMonth` (décrits dans la section 2.4.2). Ces types sont définis par le Schéma XML et repris par OWL. Ils peuvent être utilisés pour modéliser des données temporelles telles que la date de naissance d'une personne, l'instant où un événement se produit, *etc.* Par exemple, le fragment suivant définit la date de naissance de la *personne Paul.*

```
<owl:DatatypeProperty rdf:ID="dateDeNaissance">
        <rdfs:domain rdf:resource="#Personne"/>
        <rdfs:range rdf:resource="&xsd;dateTime"/>
</owl:DatatypeProperty>
<Personne rdf:ID="Paul">
        <dateDeNaissance rdf:datatype="&xsd;dateTime">
         2009-10-03T20:49:00
        </dateDeNaissance>
</Personne>
```

Il est important d'observer qu'il n'est pas possible de définir des intervalles, sauf ceux d'une durée préétablie (un jour, un mois, une année…), supportés par le type employé. Par exemple, le fragment suivant déclare le mois de Mars comme la période pendant laquelle l'entreprise Xerox publie des offres de stages d'été.

```
<owl:DatatypeProperty rdf:ID="périodeDEmbauche ">
        <rdfs:domain rdf:resource="#Entreprise"/>
        <rdfs:range rdf:resource="&xsd;gMonth"/>
</owl:DatatypeProperty>
<Entreprise rdf:ID="Xerox">
        <périodeDePublicationOffreStage rdf:datatype="&xsd;gMonth">
         --3
        </périodeDePublicationOffreStage>
</Entreprise>
```

On notera que des travaux récents [179] proposent d'éliminer de l'ensemble de types proposés par OWL, les types temporels qui définissent des intervalles et des instants récurrents. C'est le cas du type `xsd:date` qui modélise des intervalles ayant une durée d'une journée, du type `xsd:time`, qui représente une heure qui est définie pour chaque jour, du type `xsd:gMonthDay` qui représente un

jour d'un mois du calendrier grégorien qui revient chaque année, ... Selon [179], raisonner avec les instants et les intervalles récurrents est difficile en raison de leur sémantique complexe et mal définie : les récurrences sont irrégulières en raison d'exceptions comme les années bissextiles ou des sauts de seconde (ajustement d'une seconde du Temps Universel Coordonné (UTC)).

Les auteurs de [179] proposent l'utilisation exclusive du type `xsd:dateTime`, dont les valeurs peuvent être manipulées par des techniques similaires à celles définies pour les nombres. L'extension OWL 2 suit ces recommandations, et propose pour la modélisation des caractéristiques temporelles, seulement deux types de données : `xsd:dateTime` et `xsd:dateTimeStamp`. OWL 2 offre la possibilité de définir des restrictions pour ces deux types en utilisant les facettes de contrainte `xsd:minInclusive`, `xsd:maxInclusive`, `xsd:minExclusive`, et `xsd:maxExclusive`. Par exemple, la restriction suivante fait référence à tous les instants qui sont supérieurs ou égaux à l'instant représenté par `1956-01-01T04:00:00-05:00` :

```
DatatypeRestriction( xsd:dateTime
                     xsd:minInclusive
                     "1956-01-01T04:00:00-05:00"^^xsd:dateTime )
```

3.4.2 L'Ontologie OWL-Time

Des travaux récents [131, 132, 194] proposent une ontologie, appelée OWL-Time[40], pour la modélisation de phénomènes temporels complexes pour le Web Sémantique. Cette ontologie, illustrée dans la figure 3.8, est devenue recommandation du W3C pour la description du contenu temporel des pages Web et pour la description des propriétés temporelles des services Web.

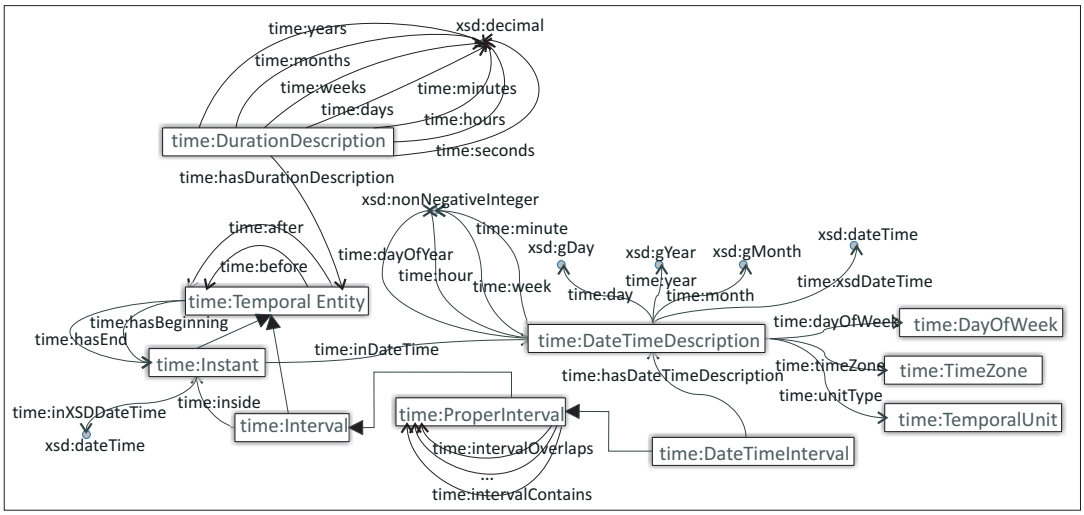


FIG. *3.8: Représentation graphique de l'ontologie OWL Time.*

L'ontologie considère deux types d'entités temporelles : les *instants* et les *intervalles*. Un instant représente un moment ponctuel dans le temps et peut être défini à l'aide de l'attribut `inXSDDateTime`. Un intervalle est défini en précisant le premier et le dernier instants qu'il inclut (par le biais des attributs `hasBeginning` et `hasEnd`). Les intervalles propres sont des intervalles particuliers, ayant des instants de début et de fin distincts.

[40] http ://www.w3.org/2006/time

Pour décrire la durée d'une entité temporelle (`TemporalEntity`), ainsi que pour pouvoir utiliser les durées comme objets à part entière et les comparer si besoin, on dispose de la classe `DurationDescription`. Celle-ci est connectée avec la classe des entités temporelles (`TemporalEntity`) par l'intermédiaire de la propriété-lien `durationOf` et possède un ensemble de huit attributs, qui décrivent la durée selon plusieurs niveaux de détail [132]. Par exemple, une durée de 5 ans, 4 mois et 3 semaines peut être définie comme suit :

```
<DurationDescription rdf:about="duration_1">
        <years rdf:datatype="&xsd;decimal">5</years>
        <months rdf:datatype="&xsd;decimal">4</months>
        <weeks rdf:datatype="&xsd;decimal">3</weeks>
</DurationDescription>
```

Une classe spéciale dans cette ontologie est `DateTimeDescription`. Elle est utilisée pour décrire des intervalles implicites, tels que "8 Mai", qui représente un intervalle de 24 heures. Toute entité temporelle a une durée, mais seulement les objets de type `DateTimeInterval` ont une description `DateTimeDescription`. La propriété-lien `unitType` spécifie l'unité temporelle la plus spécifique qui sera prise en compte dans la description. Si, par exemple, pour une description donnée, la propriété `unitType` a la valeur `unitMonth`, même si les propriétés `week`, `dayOfYear`, `dayOfWeek`, `day`, `hour`, `minute` et `second` sont définies, elles sont ignorées.

L'intérêt de cette classe vient du fait qu'elle permet de décrire un intervalle en utilisant un vocabulaire complet par rapport à celui proposé par le type `xsd:dateTime` du Schéma XML. On peut, par exemple, préciser le jour de la semaine, ou le mois qui inclut un certain intervalle (`DateTimeInterval`). Si on compare les deux descriptions suivantes, on peut dire que la représentation à travers l'utilisation du type `xsd:dateTime` est plus concise, mais la description proposée par `DateTimeDescription` est beaucoup plus expressive pour les utilisateurs.

```
<Instant rdf:about="photoShot">
        <inXSDDateTime rdf:datatype="&xsd;dateTime"
            >2009-05-17T13:01:14</inXSDDateTime>
        <inDateTime rdf:resource="photoShotTime"/>
</Instant>

<DateTimeDescription rdf:about="photoShotTime">
        <year rdf:datatype="&xsd;gYear">2009</year>
        <month rdf:datatype="&xsd;gMonth">5</month>
        <day rdf:datatype="&xsd;gDay">17</day>
        <hour rdf:datatype="&xsd;nonNegativeInteger">13</hour>
        <minute rdf:datatype="&xsd;nonNegativeInteger">01</minute>
        <unitType rdf:resource="#unitMinute"/>
        <dayOfWeek rdf:resource="#Sunday"/>
</DateTimeDescription>
```

3.4.3 Modélisation temporelle qualitative avec OWL-Time

En ce qui concerne la représentation temporelle qualitative, l'ontologie OWL-Time définit treize relations temporelles entre des intervalles propres. Ces relations - `intervalEquals`, `intervalBefore`, `intervalMeets`, `intervalOverlaps`, `intervalStarts`, `intervalDuring`, `intervalFinishes`, `intervalAfter`, `intervalMetBy`, `intervalOverlappedBy`,

`intervalStartedBy`, `intervalContains`, `intervalFinishedBy` - sont inspirées par l'algèbre d'intervalles d'Allen [14], présentée dans la section 2.4.2.

Cet ensemble est complété par deux relations qualitatives, `before` et `after`, définies pour toutes les entités temporelles. Ces dernières sont basées sur la relation d'ordre entre les instants. Si on représente le temps par une droite orientée (de gauche à droite), t, alors une entité temporelle t_1 a une relation `before` avec une autre entité temporelle t_2, si la fin de t_1 est située sur la droite t, à gauche par rapport au début de t_2.

Pour définir des relations d'inclusion entre un intervalle et un instant, on peut utiliser la propriété-lien `inside`. L'ontologie définit également la propriété-lien `inDateTime`, pour spécifier des relations d'inclusion entre un instant et un intervalle de type `dateTime`. À l'aide de cette propriété, on peut approximer un instant en spécifiant l'intervalle `dateTime` qui l'inclut. Cette facilité est très importante pour annoter les documents du Web, car elle donne la possibilité aux utilisateurs d'annoter un document (i.e. photo, document multimédia, *etc.*) d'une façon plus souple, en utilisant des informations qui sont plus aisées à mémoriser. Par exemple, lors de la publication sur le Web du compte rendu d'une réunion, pour un utilisateur qui souhaite annoter ce document avec des informations temporelles et spatiales, il est plus facile de se rappeler du fait que la réunion a eu lieu le premier lundi de ce mois que de se rappeler la date exacte.

Des caractéristiques temporelles peuvent être attachées aux objets par l'intermédiaire de la propriété-lien `holds`, comme illustré dans la figure 3.9. Elle a deux spécialisations, `atTime` et `during`, dont les définitions sont adaptées pour les instants et, respectivement, pour les intervalles.

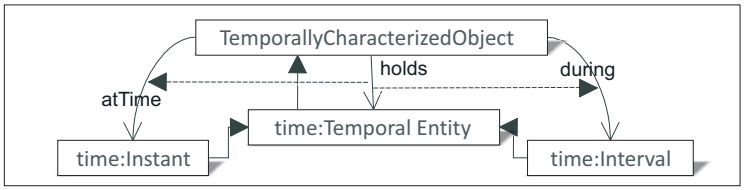


FIG. *3.9: Les relations qui font la connexion entre les objets et leurs caractéristiques temporelles.*

3.4.4 Raisonnement temporel pour OWL

Aucun des raisonneurs que nous avons étudiés (voir section 3.2.4) n'offre des inférences qui puissent exploiter de façon dédiée les informations temporelles quantitatives modélisées en OWL, en utilisant les types prédéfinis ou les concepts définis par l'ontologie de référence, OWL-Time. Or, ce type de raisonnement est très utile pour le traitement de requêtes personnalisées ou adaptées au contexte de l'utilisateur. Si on considère l'exemple d'un utilisateur qui souhaite se rendre au théâtre, et qui cherche sur le Web les spectacles qui ont lieu à proximité de chez lui, un moteur de recherche sémantique doit être en mesure de lui proposer exclusivement les spectacles à venir, et non pas les spectacles qui ont eu lieu dans le passé. Le moteur de recherche doit avoir à sa disposition des raisonneurs capables de comparer les annotations temporelles des spectacles avec la date courante. De plus, si l'utilisateur est intéressé par les spectacles qui auront lieu dans 3 jours, le raisonneur doit pouvoir réaliser des calculs à partir de données.

À notre connaissance, il n'existe pas encore de moteurs d'inférence qui puissent déduire des relations temporelles qualitatives entre les instants et les intervalles décrits en OWL. Comme décrit dans la section 2.4, ces déductions peuvent se faire à travers des calculs mathématiques si on dispose de données quantitatives, ou à l'aide d'algèbre d'intervalles si on dispose d'un ensemble

de relations qualitatives.

3.5 Synthèse

Dans la première partie de ce chapitre, nous avons présenté le Web Sémantique qui représente le contexte général de nos travaux. Après une courte présentation de l'architecture de ce Web *"nouvelle génération"*, nous avons passé en revue les travaux de standardisation qui proposent des langages pour la définition d'annotations sémantiques (RDF(S)) et d'ontologies (OWL et OWL 2).

Nous avons notamment constaté les limites de l'expressivité du langage RDF(S) qui ont conduit à la définition d'un langage plus complexe, OWL, décliné en trois sous-langages d'expressivité croissante, OWL Lite, OWL DL et OWL Full, qui répondent aux besoins variés des applications. Nous avons également présenté certains outils pour la modélisation d'ontologies ainsi que plusieurs outils d'inférence. Il est important de souligner que les raisonneurs, pour la plupart conçus pour les *Logiques de Description*, offrent un support très limité pour des calculs algébriques.

La deuxième partie du chapitre est centrée sur le Web Sémantique Géospatial, qui regroupe les activités autour du développement d'ontologies spatio-temporelles pour la description sémantique d'informations à références spatiales et temporelles accessibles sur le Web. Après avoir noté le déficit des langages à ontologies en terme de types spatiaux et temporels mis à disposition, nous avons étudié les différentes approches de modélisation des annotations spatiales et temporelles quantitatives et qualitatives à travers des concepts et des relations définis par des ontologies spécialisées. Dans cette direction, nous avons présenté l'ontologie *GeoRSS-Simple*, devenue recommandation du W3C comme vocabulaire de référence pour la description des propriétés géospatiales des ressources Web. Nous avons surtout constaté l'absence d'une ontologie de référence recommandée par W3C pour la description des relations spatiales qualitatives.

Concernant la représentation des annotations temporelles quantitatives, les langages d'ontologies mettent à disposition un ensemble de types dédiés. Pour la description des relations temporelles, on peut se rapporter à l'ontologie *OWL Time*, recommandation du W3C pour la modélisation des concepts temporels et qui propose également une modélisation pour les treize relations l'Allen (voir chapitre 2).

Si plusieurs approches pour la modélisation des informations spatiales et temporelles existent à ce jour, il faut souligner le manque (et le besoin) de raisonneurs spatiaux et temporels capables d'exploiter ces données pour déduire des informations implicites ou/et pour répondre à des requêtes.

Ainsi, les informations spatiales et temporelles restent très peu exploitées par les moteurs de recherche, principaux outils d'exploitation du Web, alors qu'elles pourraient apporter plus de précision, permettre d'affiner une requête, de limiter la recherche en utilisant des contextes temporels et spatiaux, *etc.*

Chapitre 4

Exploration du Web Sémantique (Géospatial)

La vraie puissance du Web n'est pas le lien comme tel mais plutôt les liens implicites qui relient l'information.

Michael Jones, Chief technologist de Google Earth

SOMMAIRE

Résumé

*Ce chapitre explore les techniques d'exploitation du Web et du Web Sémantique qui existent à ce jour. Il décrit également un nouveau paradigme de recherche, l'**analyse sémantique**, dont le but est d'exploiter les ontologies du Web Sémantique pour découvrir les connexions qui existent entre les individus. Cette étude présente les différents types de connexion qui peuvent exister entre des individus, leur sémantique formelle, les opérateurs qui permettent leur découverte, ainsi que quelques critères de classement.*

4.1 Introduction

Comme nous l'avons montré dans le chapitre 3, le Web Sémantique [34] vise à offrir des solutions pour accroître la performance et le rappel des moteurs de recherche, en annotant le contenu des ressources Web à l'aide de concepts ontologiques compréhensibles et exploitables par les machines. Autrement dit, le but de ce nouveau Web est d'associer une couche descriptive aux pages classiques, afin de rendre possible une exploitation plus efficace et automatisée du Web.

Après avoir décrit dans le chapitre précédent les langages de définition d'ontologies, ainsi que les outils de modélisation et d'inférence qui existent à ce jour, dans ce chapitre nous présentons plusieurs techniques de fouille de données contenues par les pages Web. L'objectif est de proposer des méthodes pour *i)* la création semi-automatique d'ontologies à partir des documents Web, et *ii)* pour l'exploitation des connaissances contenues par ces ontologies dans le but de répondre à des requêtes.

Nous nous concentrons surtout sur l'*analyse sémantique* des graphes RDF(S). Ce nouveau paradigme de recherche vise la découverte des liens directs et indirects qui existent entre différents individus, décrits dans des graphes RDF(S). Notre objectif, par la suite, est d'appliquer les techniques d'*analyse sémantique* pour l'exploitation des connaissances du Web Sémantique Géospatial.

4.2 Les techniques de fouille

Sur le Web, des données ou des documents sont disponibles en quantité (très) importante et en qualité (très) variable, sans utilisation entendue particulière et précise [185]. De plus, la caractéristique majeure du Web est sa nature extrêmement dynamique. Ceci a un réel impact sur la construction et la maintenance des ontologies qui sont confrontées de façon continue à un problème d'évolution. Étant donnés les niveaux de complexité qui peuvent être atteints par les ontologies, un processus, au moins semi-automatique de maintenance s'impose de plus en plus pour faciliter cette tâche et assurer sa fiabilité [98]. Actuellement, il n'existe pas de consensus sur les méthodes proposées et des lignes de conduite à adopter pour un tel processus [98]. Cependant, les textes sont reconnus comme étant une source essentielle à l'élaboration des ontologies et des connaissances du domaine[98].

4.2.1 Fouille du Web

La *fouille de données* est un ensemble de méthodes et techniques dédiées à l'extraction de connaissances à partir de données complexes et volumineuses [23]. Appliquée au Web, elle s'attache à explorer et exploiter le contenu, l'usage et la structure des documents, que ce soit des pages HTML, des documents multimédias, des liens hypertextes, des fichiers logs ou bien d'autres données générées suite à l'utilisation du Web. Cela peut aider à la découverte de structures globales aussi bien que locales (*i.e. motifs* ou *patrons*) à l'intérieur et entre les pages Web [31]. La fouille du Web peut tirer un grand avantage de l'exploitation de la structure des données (*i.e.* la structure des tables de bases de données, les schémas XML, la structure des graphes RDF(S) ou d'ontologies...), mais peut également s'appliquer à des données semi-structurées ou non structurées comme le texte libre [31]. Ainsi, la fouille du Web constitue un appui inestimable pour la construction des ontologies [31].

Selon Kosala et Blockeel [141], les techniques de fouille de données sur le Web peuvent être classées en trois catégories : *(i)* fouille du *contenu* du Web, *(ii)* fouille de la *structure* du Web, et *(iii)* fouille de l'*usage* du Web.

Fouille du contenu du Web

La fouille du contenu est relativement similaire à la fouille de textes, et fait référence à la découverte des connaissances à partir du contenu textuel de pages Web [23]. Il s'agit notamment de détecter des co-occurrences de termes dans les textes des pages Web. Par exemple, les co-occurrences de termes dans des articles présents dans des flux RSS de nouvelles peuvent montrer que le terme *"or"* est souvent évoqué conjointement au terme *"cuivre"*, lorsque les articles concernent le Canada, et, au terme *"argent"* lorsque les articles concernent les États-Unis. Il est également possible de découvrir des évolutions ou des tendances, qui indiquent une augmentation ou diminution de l'intérêt pour certains sujets [31]. Un autre domaine d'application est la détection d'événements. Par exemple, l'identification de nouveaux événements ou d'événements non identifiés auparavant à partir de flux RSS de nouvelles.

Selon [185], il existe deux types principaux de méthodes de fouille de données qui peuvent s'appliquer à la fouille du contenu Web :

a) les méthodes symboliques qui incluent la classification par treillis, par arbres de décision, classification conceptuelle, recherche de motifs fréquents, extraction de règles d'association, les méthodes d'apprentissage (*i.e.* l'induction, apprentissage à base d'instances, apprentissage à base d'exemples, *etc.*), les méthodes de recherche d'information et de réponse aux questions, *etc.*

b) les méthodes numériques : statistiques, analyse des données, modèles de Markov cachés d'ordre 1 et 2, réseaux bayésiens, réseaux de neurones, et algorithmes génétiques, *etc.*

Fouille de la structure du Web

La fouille de la structure opère sur les hyperliens et peut être vue comme une spécialisation de la fouille du contenu [23]. Cette technique d'analyse du Web utilise comme ressource principale un ensemble de pages Web (*i.e.* qui peut inclure un seul site Web ou le Web en entier), et exploite les liens hypertextes qu'elles contiennent, ainsi que l'information implicite portée par ces *hyperliens* (*i.e.* des relations de similarité entre les sujets abordés par différentes pages) [31]. Par conséquent, une application importante est l'identification de la pertinence relative des pages ayant une pertinence équivalente, calculée suite à l'analyse de leur contenu isolé [31].

Par exemple, la recherche par sujet déduit à partir d'hyperliens (en anglais *hyperlink-induced topic search*) consiste en l'analyse de la topologie des hyperliens contenus par le Web, dans le but de découvrir les sources d'informations faisant état d'autorité par rapport à un sujet de recherche donné. Ce genre d'information est recensé par les *pages* dites d'*autorité*, définies en relation avec les *hubs* [31]. Le moteur de recherche Google, par exemple, doit son succès à l'algorithme *PageRank*, qui stipule que la pertinence d'une page par rapport à un sujet *S*, augmente avec le nombre de liens hypertextes désignant la page respective dans d'autres pages, et, en particulier, dans des pages connues comme étant pertinentes par rapport à *S* [191].

Fouille de l'usage du Web

L'analyse de l'usage du Web étudie principalement les enregistrements des demandes d'accès faites par les visiteurs d'un site Web, le plus souvent recensées dans les logs de serveur [31]. Si le contenu et la structure des pages Web, et notamment ceux des sites Web, reflètent les intentions de leurs concepteurs et développeurs, ainsi que l'architecture et l'organisation des données présentées dans les pages Web, les informations sur la navigation et l'exploitation des pages Web par les utilisateurs/agents logiciels peuvent également offrir des pistes à propos de la structuration de l'information [31].

Tout d'abord, pour les pages Web non-structurées, certaines relations peuvent être déduites entre les différentes parties/éléments de la page Web, à partir de modèles de navigation identifiés dans les fichiers logs. Par exemple, les catalogues de produits en ligne ne sont, en général, pas très bien structurés (*i.e.* les produits différents sont mélangés dans une organisation de type ensemble), ou disposent d'une organisation hiérarchique (donnée par les catégories de produits, les fabricants, *etc.*) basique [31]. Néanmoins, la fouille de l'usage d'un tel site peut faire ressortir des relations de type : la plupart des utilisateurs (*i.e.* 80 %) qui se sont intéressés au produit *A* ont également regardé le produit *B* [31]. Dans ce cas, l'intérêt de l'utilisateur est mesuré par le biais des demandes d'accès à la page de description des produits respectifs, ou par l'ajout des produits dans le panier d'achat, *etc.* L'identification de ce type de règles d'association est à la base de plusieurs stratégies de vente dans le commerce électronique (*cross-selling*, *up-selling etc.*) [31]. Concrètement, lorsqu'un nouvel utilisateur montre de l'intérêt pour un produit *A*, il reçoit automatiquement une recommandation pour le produit *B*.

4.2.2 Fouille de la sémantique du Web pour la construction du Web Sémantique

Nous rappelons que le Web Sémantique preconise d'ajouter des annotations sémantiques aux ressources Web afin de permettre la manipulation et l'exploitation automatique des connaissances que celles-ci contiennent. Dans cette direction, la fouille du Web peut offrir une aide importante en ce qui concerne l'apprentissage et la construction semi-automatique d'ontologies [31]. Les méthodes de fouille de la sémantique que nous présentons dans la suite sont des outils semi-automatiques qui assistent les ingénieurs dans le processus d'extraction de connaissances, mais qui ne peuvent pas remplacer l'intervention d'experts humains en totalité, car les données analysées contiennent souvent des références implicites à des connaissances externes (*i.e.* conventions sociales, contexte implicite, connaissances issues de l'expérience, *etc.*).

Apprentissage d'ontologies[1]

L'apprentissage d'ontologies est basé sur l'intégration de nombreuses disciplines de recherche liées à la fouille de données, afin de faciliter la construction semi-automatique d'ontologies. Il s'agit en particulier du domaine de l'ingénierie de la connaissance, et de l'apprentissage automatique [161].

Maedche et Staab [160] classent les différentes approches d'apprentissage d'ontologies en fonction du type de ressources exploitées : *a)* apprentissage à partir de textes, *b)* apprentissage à partir de dictionnaires, *c)* apprentissage à partir de bases de connaissances, *d)* apprentissage à partir de données semi-structurées, *e)* apprentissage à partir de schémas relationnels.

[1] En anglais *ontology learning*, *ontology extraction*, *ontology generation* ou *ontology acquisition.*

a) **La construction d'ontologies à partir de textes** est basée sur l'application de techniques d'analyse du langage naturel[2] pour le texte écrit [103]. Les approches les plus connues de ce type sont :

- l'*extraction de motifs*, qui base la reconnaissance d'une *relation* entre entités sur l'occurrence, dans le texte étudié, d'une séquence de mots qui respectent un modèle/patron donné [103] ;
- l'*extraction de règles d'association*, une technique utilisée dans le processus de fouille de données pour découvrir des informations stockées dans des bases de données, est applicable dans le cas où on a une idée générale du type de relations qu'on souhaite découvrir [103] ;
- la *classification de concepts*[3], qui organise les concepts dans des hiérarchies. La classification se fait en fonction de la distance sémantique qui existe entre les concepts. La méthode de calcul de la distance sémantique entre concepts dépend de différents facteurs et doit être fournie par l'utilisateur ;
- le *raffinement d'ontologies*[4] qui vise la construction d'une *ontologie de domaine* à partir de sources de données hétérogènes [103]. Selon Gómez-Pérez et Manzano-Macho [103], ce processus a plusieurs étapes : dans un premier temps, un noyau ontologique général est utilisé comme structure générique de haut niveau pour l'ontologie de domaine. Dans un deuxième temps, un dictionnaire contenant les termes principaux du domaine, décrits en langage naturel, est utilisé pour faire l'acquisition de concepts du domaine. Ces concepts sont ensuite classés par rapport au noyau ontologique. Finalement, des corpus de texte généraux et spécifiques au domaine analysé sont utilisés afin d'éliminer les concepts qui ne sont pas spécifiques au domaine. Cette dernière phase repose sur la théorie selon laquelle les concepts spécifiques à un domaine D sont plus fréquents dans un corpus spécifique à D que dans un corpus général ;
- l'*apprentissage de concepts* consiste à mettre à jour de façon incrémentale une taxonomie de concepts, au fur et à mesure que des nouveaux concepts sont découverts suite à la fouille de textes [103].

b) L'**apprentissage d'ontologies à partir de dictionnaires** est basé sur l'utilisation de dictionnaires compréhensibles par les machines pour l'extraction de concepts et des relations les liant [103].

c) L'**apprentissage d'ontologies à partir de bases de connaissances** vise la construction d'ontologies à partir de bases de connaissances existantes [103].

d) L'**apprentissage d'ontologies à partir de données semi-structurées** vise l'exploitation de la structure des documents (*i.e.* définie en utilisant le XML Schéma) pour la construction d'ontologies [103].

e) L'**apprentissage d'ontologies à partir de schémas relationnels** consiste en l'extraction de concepts et relations pertinentes, à partir de connaissances stockées dans des bases de données.

[2] Connues aussi sur le nom de Traitement du Langage Naturel (TAL).
[3] En anglais *concept clustering.*
[4] En anglais *ontology pruning.*

Peuplement d'ontologies[5]

La mise en place du Web Sémantique ne peut pas se baser sur une annotation manuelle réalisée par les utilisateurs, car celle-ci est répétitive, coûteuse, et consommatrice de temps. De plus, convaincre les utilisateurs d'Internet d'annoter en utilisant des ontologies les documents Web qu'ils produisent, est difficile et nécessiterait une action au niveau mondial et une prise de conscience collective universelle qui n'a pas encore eu lieu. D'ailleurs, l'annotation statique associée à un document peut être inachevée, incorrecte et/ou peut ne pas prendre en compte les mises à jour des pages. Il est donc important de disposer de méthodes pour l'annotation automatique/semi-automatique des pages [47].

Sur cette voie, le peuplement d'ontologie est un processus d'enrichissement de la partie assertionnelle d'une ontologie (la ABox) avec de nouvelles instances de concepts, d'attributs et de relations, telles que définies dans la terminologie d'une ontologie de référence (qui peut être l'ontologie enrichie). Ce processus est basé sur des fonctionnalités offertes par les technologies du Traitement du Langage Naturel (TLN), et notamment celles de l'Extraction d'Informations. Le TLN est également utilisé, dans le cadre du Web Sémantique, pour la construction semi-automatique de vocabulaires/terminologies d'un domaine à partir d'un corpus documentaire représentatif, ainsi que pour leur maintenance [40], pour l'enrichissement semi-automatique de bases de connaissance par les entités nommées et les relations sémantiques extraites des documents textuels après validation [138], pour l'annotation sémantique semi-automatique de ressources documentaires [125], *etc.*

Bien que faisant déjà l'objet de nombreuses recherches, ces tâches d'annotation sémantique et de peuplement d'ontologies restent un véritable challenge, pas seulement dans le domaine de l'ingénierie logicielle mais quel que soit le domaine étudié. En effet, le rôle des humains demeure bien souvent irremplaçable et l'automatisation reste un des plus grands besoins pour de tels outils, particulièrement lorsqu'il s'agit d'annoter de grandes collections de documents [231].

Alignement et fusion d'ontologies

Dans de nombreux contextes applicatifs, tels que l'échange de documents sur le Web, la médiation à travers des sites Web, la communication des agents logiciels sur Internet, l'intégration des services Web, la réécriture de requêtes, *etc.*, plusieurs ontologies couvrant, totalement ou partiellement, un même domaine se trouvent impliquées [240]. Pour permettre l'interopérabilité des applications et/ou agents qui s'appuient sur une de ces ontologies, l'hétérogénéité entre les connaissances exprimées au sein de chacune d'entre elles doit être résolue [91]. À cette fin, il est nécessaire d'établir des liens sémantiques entre les entités appartenant à deux ontologies différentes, par le biais d'un processus appelé alignement d'ontologies.

Étant données deux ontologies, l'alignement produit un ensemble de correspondances, chacune liant deux entités (par exemple, des concepts, des instances, des propriétés, des termes, *etc.*) par une relation (équivalence, subsomption, incompatibilité, *etc.*), éventuellement munie d'un degré de confiance [91]. L'ensemble des correspondances, aussi appelé *alignement*, peut par la suite être utilisé pour fusionner les ontologies, échanger des données entre ontologies, ou traduire des requêtes formulées sur une ontologie vers une autre [79].

Selon Kengue *et al.*, la confrontation deux-à-deux des entités qui a pour but de leur affecter

[5] En anglais *instance learning*.

un degré "d'alignabilité", peut être réalisée de nombreuses façons et en se basant sur des aspects variés [79]. Par exemple, pour deux classes OWL, une ou plusieurs dimensions descriptives, telles que leurs noms, leurs instances connues, les super-classes respectives, les listes d'attributs, les propriétés-lien, *etc.*, peuvent être choisies. Toutes les méthodes d'alignement ont en commun le fait qu'elles déterminent des correspondances entre les entités ontologiques en utilisant des mesures de similarité [240]. En reprenant les mesures de similarité présentées par Rahm et Bernstein [206], Zghal *et al.* proposent dans [240] la classification suivante :

a) *La méthode terminologique* qui consiste en la comparaison des labels des entités. Elle se divise en deux approches, l'une purement syntaxiques, l'autre lexicale. L'approche syntaxique effectue la correspondance à travers les mesures de dissimilarité des chaînes de caractères (*i.e.* EditDistance), tandis que l'approche lexicale effectue la correspondance à travers les relations lexicales (*i.e.* synonymie, hyponymie, *etc.*) ;

b) *La méthode de comparaison des structures internes* consiste en une comparaison des attributs des entités (*i.e.* intervalle de valeur, cardinalité d'attributs, *etc.*) ;

c) *La méthode de comparaison des structures externes* compare les relations d'une entité avec les relations d'autres entités ;

d) *La méthode de comparaison des instances* compare les extensions des entités de type classe, propriété et/ou type de donnée (*i.e.* l'ensemble des instances/tuples[6] attachées aux classes/propriétés visées par la comparaison) ;

e) *La méthode sémantique* compare les interprétations (ou plus exactement les modèles) des entités.

La définition d'une mesure de similarité entre entités sera exprimée comme la combinaison (par exemple, dans une somme ou une combinaison linéaire) des similarités de chacune des dimensions descriptives choisies pour l'alignement [79]. Par exemple, pour les classes, on peut choisir de ne travailler que sur les noms et les attributs, ignorant les rôles et les super-classes éventuelles. La comparaison sur chacune des dimensions requiert une fonction dédiée. Dans le cas des noms des entités, de nombreuses possibilités existent dont la plus simple est la comparaison de ceux-ci en tant que chaînes de caractères [79]. Cependant, une telle comparaison risque, selon Kengue *et al.*, d'être parfois trompeuse comme le montre l'exemple de `Person` et `Human`. Une approche alternative consiste à utiliser un thésaurus ou autre ressource lexicale afin de capter les rapports entre termes (synonymie, homonymie, hypo-/hypero-nymie, *etc.*).

4.2.3 Fouille du Web Sémantique

4.2.3.1 Recherche d'ontologies

La recherche d'ontologies utiles et pertinentes pour appuyer des tâches spécifiques est l'un des défis principaux sur le chemin vers l'intégration et la réutilisation efficace des domaines modélisés par les différentes communautés. Dès l'élévation de OWL au rang de langage standard pour la définition des ontologies, des libraires ontologiques contenant des ontologies prêtes à l'emploi, proposées par diverses communautés de recherche, ou même par des internautes, ont été mises à disposition sur l'Internet. On peut citer de façon non exhaustive : "DAML Ontology Library"[7]

[6] Dans ce travail, nous utilisons le terme *tuple* pour désigner l'*instance* d'une relation (ou propriété-lien).

[7] http ://www.daml.org/ontologies/

ou "Protégé Ontologies Library"[8]. Cependant, même si on dispose ainsi d'un nombre croissant d'ontologies largement accessibles pour toute application, choisir "la bonne" ontologie pour une tâche spécifique reste un problème.

Les moteurs de recherche sémantique existants, comme Swoogle [78] ou OntoKhoj [196] retrouvent les ontologies contenant certains mots clé, mais n'offrent pas la possibilité de raffiner les requêtes en spécifiant les relations (subsomption, association, propriétés-attributs) caractérisant les concepts visés. Thomas *et al.* proposent un moteur de recherche plus puissant, ONTOSEARCH2 [229], qui s'appuie sur deux composantes principales : un entrepôt d'ontologies et un moteur de recherche. En mémorisant des traductions approximatives pour des ontologies OWL en DL-Lite [46], ONTOSEARCH2 permet l'interrogation de son entrepôt, soit en totalité, soit en partie, à travers des requêtes SPARQL. Il intègre différents niveaux d'interrogation des libraires ontologiques : un outil de recherche syntaxique, un outil répondant aux requêtes sémantiques et un outil de recherche basé sur la logique floue f-DL-Lite. L'emploi des traductions DL-lite pour les ontologies OWL vise l'augmentation de l'efficacité en temps de réponse du système, en s'appuyant sur l'idée de compilation des connaissances. L'utilisateur décrit son domaine ontologique dans un langage de représentation expressif et puissant qui répond aux exigences imposées par la majorité des applications pratiques, et ses déclarations seront "compilées" dans un langage moins expressif mais qui admet des inférences efficaces et rapides.

Pour la construction de son entrepôt d'ontologies, ONTOSEARCH2 établit des *mapping* (correspondances) entre les concepts définis dans les ontologies soumises, et les méta-données qui les caractérisent. Les méta-données ont des pondérations différentes qui varient entre `rdfs:label` (le plus fort) et `rdfs:comment` (le plus faible), en passant par l'URL associé aux ressources. Ces mots clé sont hérités à travers la hiérarchie de concepts ainsi qu'à travers la hiérarchie de propriétés. Les annotations définies au niveau de l'ontologie seront aussi valables au niveau des concepts et des instances qu'elle inclut. Les requêtes sont exprimées, soit à travers une interface d'interrogation textuelle, soit comme des requêtes SPARQL qui peuvent éventuellement faire référence à des extensions floues pour modéliser le degré de confiance attendu pour chaque terme recherché. De plus, la requête peut spécifier la partie de l'ontologie qu'elle vise : les concepts, les propriétés, les ressources ...

4.2.3.2 Moteurs de recherche sémantiques

Suite à l'essor des langages de modélisation d'ontologies et des outils d'annotation sémantique, de nouveaux moteurs de recherche permettront bientôt de répondre aussi bien à des requêtes génériques, du type : "quelles sont les publications dans le domaine de l'Informatique ?" qu'à des requêtes beaucoup plus fines, croisant le contenu de plusieurs documents hétérogènes, comme par exemple "quelles sont les publications qui parlent de Web Sémantique et dont l'un des auteurs travaille à Grenoble" [216]. La recherche d'information sortira, ainsi, de la seule recherche morphologique, par comparaison de chaînes de caractères, pour aborder la "recherche intelligente" sur les contenus. Dans cette direction, plusieurs moteurs de recherche pour le Web Sémantique ont déjà été développés, avec le but d'appuyer la recherche sémantique d'information, par la manipulation formelle des ontologies et des annotations. Parmi ceux-ci, on peut citer SPIRIT[227], CORESE [69], ZOOM[9], Swoogle [77], *etc.*

[8] http ://protegewiki.stanford.edu/index.php/Protege_Ontology_Library
[9] http ://www.acetic.fr/zoom.htm

CORESE est un moteur de recherche qui permet de retrouver des ressources qui ont été préalablement décrites au moyen d'une ontologie. La recherche se fait en exploitant les connaissances représentées dans l'ontologie et les annotations portant sur le contenu sémantique des documents. CORESE implémente le langage RDF sous forme de graphes conceptuels [69]. Il comprend également un langage de règles et un moteur d'inférence associé, qui permet d'augmenter et de valider les descriptions de ressources [69].

D'autres moteurs de recherche, tels que SPIRIT, utilisent les ontologies dans le processus d'extension/ reformulation de la requête. Différents termes peuvent servir à exprimer une même idée. C'est pour cela qu'il est souvent utile, selon [227], de pouvoir transformer/reformuler une requête en des termes sémantiquement équivalents afin d'étendre la recherche dans la base d'index. Le module de reformulation de SPIRIT, par exemple, traite une interrogation en langage libre, que l'utilisateur spécifie à travers l'interface d'interrogation. Son dictionnaire standard de règles de reformulation comprend, pour une langue donnée, des associations entre termes de même famille et synonymes. Par exemple, l'utilisateur posant la requête "l'élaboration du Plan d'Assurance Qualité" retrouvera les documents comportant le texte "le Plan d'Assurance Qualité pourra être élaboré par ...", "élaboration" et "élaboré" étant des mots de la même famille. À partir de l'analyse des mots de la requête, les termes proposés par le module de reformulation sont issus d'une double opération :

- de la recherche de l'ensemble des termes associés présents dans le dictionnaire de reformulation,
- du recoupement effectué entre les termes trouvés et les index présents dans la base.

Les ontologies peuvent également être utilisées pour l'indexation des documents, le filtrage de résultats et la visualisation de résultats [184].

4.3 Analyse Sémantique - nouveau paradigme d'interrogation du Web Sémantique

Nous avons vu dans les sections précédentes que, afin de pouvoir exploiter les annotations sémantiques, les moteurs de recherche actuels du Web doivent évoluer. La direction de développement la plus évidente est d'orienter la recherche vers la découverte des ressources dont la description sémantique est compatible avec la sémantique des termes de la requête. Par exemple, pour une requête de type "*Chercheur dont le nom est Philippe Lamarre*", l'ensemble des résultats doit contenir exclusivement les documents qui font référence à une instance du concept *Chercheur*, dont le nom est *Philippe Lamarre*, et non pas tous les documents qui contient les termes *Chercheur*, *Philippe* et *Lamarre*.

Toutefois, si cette problématique est de toute évidence très importante, un défi complémentaire par rapport aux approches présentées dans la section précédente, serait d'orienter les moteurs de recherche vers la découverte, non pas des documents intéressants, mais des relations explicites et/ou implicites entre les entités décrites par ces documents. Ce type d'analyse est très utile pour soutenir, par exemple, la prise de décision dans des domaines tels que la détection des conflits d'intérêt ou la sécurité d'un territoire. Dans ce domaine, l'analyste est intéressé premièrement par les relations qui existent entre les différents individus, et seulement dans un deuxième temps, par les documents qui décrivent ces relations.

Sheth *et al.* citent plusieurs défis associés à la mise en place d'une recherche sémantique basée

sur les relations entre individus [219]. Ainsi, selon les auteurs, si la recherche “classique” à base d’entités génère un très grand nombre de résultats, on peut imaginer que les techniques de recherche à base de relations vont générer un ensemble plus large encore de résultats, parmi lesquels un pourcentage très important de bruit. En général, chaque page Web contient plusieurs références à des concepts, des individus et/ou des relations, faisant partie du domaine décrit par la page en question. Ainsi, dans le Web Sémantique, chaque page Web sera décrite par plusieurs annotations qui correspondent aux concepts, relations et individus qu’elle cite. Si le nombre d’individus annotés dans le cadre du Web Sémantique est important, il est évident que le nombre de relations ou de chemins sémantiques qui relient les individus, sera beaucoup plus important encore. Un exemple très illustratif de cette idée est la théorie des six degrés de séparation. Celle-ci dit que, chacun d’entre nous peut être lié à n’importe quelle autre personne sur Terre via six personnes intermédiaires. Eric Horvitz et Jure Leskovec, chercheurs chez Microsoft, ont confirmé cette théorie, suite à une étude réalisée sur 30 milliards de messages instantanés envoyés à travers le monde par près de 250 millions de personnes en juin 2006 [150]. Ils ont réussi à mettre en évidence que deux individus pris au hasard sur le globe peuvent en moyenne être reliés en 6,6 étapes. Ainsi, si on considère le Web comme un vaste réseau social, on aura approximativement deux milliards d’individus [10] connectés entre eux par des chaînes de six individus intermédiaires en moyenne. Si on réduit le raisonnement aux relations de type `connaît` les principales relations prises en compte par les réseaux sociaux, on obtient comme nombre de relations possibles : $C^{6}_{1.600.000.000}$. Mais on peut aussi établir d’autres types de relations entre les personnes, le nombre de relations peut donc être encore plus important.

Il est aussi nécessaire, lors de la recherche d’une relation entre deux individus, d’établir des filtres pour limiter le bruit dans l’ensemble de résultats, et pour mieux cibler la recherche en fonction des préférences et des attentes de l’utilisateur.

Pour la découverte de relations complexes liant les individus sur le Web, il faut proposer de nouvelles techniques de recherche basées sur la sémantique des annotations, et sur un ensemble de déductions logiques. Une telle approche est l’*analyse sémantique* [219]. Cette dernière a pour but l’identification automatique d’*associations sémantiques* reliant deux individus x et y, au sein d’un graphe RDF(S), en vue de répondre à des questions du type : "*l’entité x est-elle reliée (même non directement) à l’entité y ?*". Les différents types d’objets sont reliés de façons complexes et souvent inattendues, d’où l’intérêt de l’analyse sémantique qui offre de nouvelles perspectives sur la découverte de connexions entre individus sans lien *a priori* apparent. Ce nouveau paradigme de recherche permet également d’associer un *contexte* à chaque requête, pour mieux préciser l’intérêt de l’utilisateur, mais aussi pour limiter le bruit (les résultats non pertinents).

4.3.1 Associations sémantiques pour RDF(S)

De nombreuses applications exigent une notion de relation plus complexe que les simples connexions directes entre les entités [219]. Par exemple, dans le cadre de la sécurité aérienne, il est devenu pertinent de permettre aux agents de sécurité dans les aéroports d’effectuer des investigations sur les liens existants entre différents passagers, du type : " quelle *relation importante* devrait exister entre le passager x et le passager y" ?. Dans ce contexte, différentes relations indirectes peuvent émerger lors de l’exploitation des relations transitives liant des personnes, des organisations, *etc.*, entre elles. Dans ce scénario, comme les relations ne sont pas connues *a priori*,

[10] Selon le site http ://www.internetworldstats.com/ le nombre d’utilisateurs de l’Internet s’élève à 1.6 milliards.

il n'est pas possible de les encoder à travers des règles. En revanche, elles peuvent être découvertes par le biais d'un processus analytique. En général, ces relations apparaissent comme un ensemble de liens ou connexions entre les entités. À titre d'exemple, considérons deux passagers qui sont citoyens du même pays, qui ont acheté leurs billets en utilisant la même carte de crédit, alors qu'ils ne font pas partie de la même famille et, entre outre, l'un d'eux est connu comme ayant des liens avec une organisation terroriste. Ces deux passagers pourrait apparaître comme fortement suspects pour la sécurité civile (ou aerienne) d'un État...

L'identification de ce genre de relations implique l'évaluation de l'ensemble des chemins (de connexion) reliant les entités les unes aux autres et qui soient pertinents du point de vue du contexte de la recherche. Une telle étude permet d'identifier de nouvelles relations inférées sur la base de la connectivité ou de la similarité des chemins.

L'analyse sémantique est un nouveau paradigme de recherche proposé par [18, 219, 19, 124, 218, 199], qui permet l'identification des relations complexes et indirectes qui relient des individus décrits dans des graphes RDF(S). Les auteurs cités nomment ce genre de relation, *association sémantique*. Les *associations sémantiques* sont définies au-dessus du modèle de graphe fournit par RDF(S).

L'*analyse sémantique* a été définie pour des graphes RDF(S) pour deux raisons. Tout d'abord, les représentations à base de graphes peuvent capturer de façon naturelle, la connectivité entre les entités. Cela permet la transformation des requêtes qui interrogent la connectivité entre deux individus en des opérations de construction de chemins, typiques pour des graphes. Cette technique est différente de l'extraction de données qui utilise des méthodes statistiques pour trouver des occurrences des relations sur la base des différents motifs identifiés dans l'ensemble de données. Ensuite, la définition du langage RDF(S), comme standard de description des annotations Web, garantit l'ouverture et la généralité de l'approche. De plus, en exploitant la sémantique formelle attachée aux déclarations RDF(S), l'approche assure des réponses pertinentes et adaptées aux besoins de l'utilisateur.

Afin d'illustrer l'utilité et la sémantique des différents concepts définis par l'*analyse sémantique*, nous utilisons le graphe RDF(S) simple présenté dans la figure 4.1. Il décrit des concepts, des relations, et des entités faisant partie du domaine de l'art. Concrètement, il s'agit de trois catégories génériques de concepts : les `Artistes`, les `Artefact` et les `Musées` ainsi que de quelques relations qui existent entre eux : `crée`, `exposé`, `peint`, *etc.* La partie supérieure de la figure 4.1 décrit le schéma RDF(S) (ou la TBox), alors que la partie inférieure illustre l'ensemble de faits ou d'assertions (ou ABox). Par la suite, nous nous intéressons à la découverte des liens qui existent entre les différentes instances de la ABox.

Le concept d'*association sémantique* entre deux individus x et y est construit à partir de la notion de *séquence de propriétés*. Cette dernière est un chemin théorique, défini au niveau du Schéma RDF (TBox), par une chaîne de propriétés (voir définition 4.1). La construction des *séquences de propriétés* pour la découverte des *associations sémantiques* est guidée pour relier la classe X à laquelle l'individu x appartient à la classe Y, dont l'individu y est une instance. Les instances concrètes d'une *séquence de propriétés*, appelées $\rho - paths$, sont définies par rapport à l'ensemble d'instances (ABox) décrit dans le graphe RDF(S) considéré. Un $\rho - path$ contient une séquence de *tuples* (instances des propriétés RDF(S)) reliant des *individus* (instances de classes RDF(S)), et qui satisfont les contraintes énoncées dans la proposition 4.1.

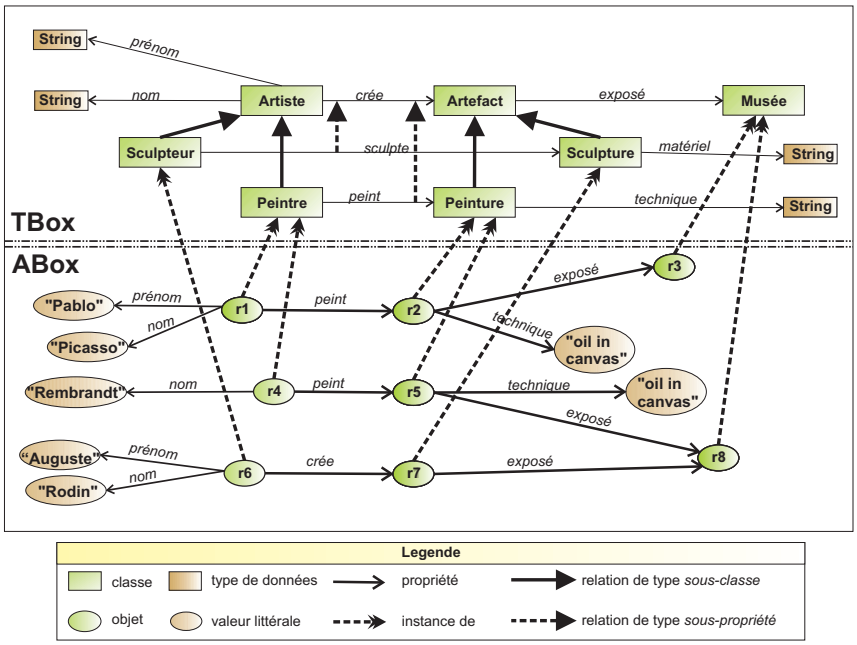


FIG. *4.1: Exemple de graphe RDF(S) simple pour l'illustration du concept d'association sémantique (source [19]).*

Définition 4.1. *Une séquence de propriétés, S, est un ensemble ordonné et fini d'éléments,* $S = \lfloor P_1 \bullet P_2 \bullet P_3 \bullet \ldots \bullet P_n \rfloor$*, où chaque* P_i *est une propriété définie par le Schéma RDF de référence R [19]. L'interprétation d'une séquence de propriétés est définie comme extension de la fonction d'interprétation d'un graphe RDF(S) (voir définition 3.1) et consiste en un ensemble de séquences de tuples, appelés* $\rho - paths$ *:*

$$I(S) = \{[\langle a_1, b_1\rangle, \langle a_2, b_2\rangle \ldots \langle a_n, b_n\rangle] \,|\, \langle a_i, b_i\rangle \in IEXT(P_i)\}, \forall i \in 1..n$$

Proposition 4.1. *Une séquence de propriétés* $S = \lfloor P_1 \bullet P_2 \bullet P_3 \bullet \ldots \bullet P_n \rfloor$ *est réalisable par rapport au graphe RDF(S) de référence R, s'il existe au moins une interprétation I de R, dans laquelle S est satisfaite (détient au moins une instance) [19] :*

$$\forall P_i \in S, \exists (x_i, y_i) \in IEXT(P_i) \,|\, y_i = x_{i+1}, \forall i \in [1, n-1].$$

Cela veut dire qu'il existe au moins une instance α de S (appelée $\rho - path$), dans la forme d'une séquence de tuples de type $\alpha = [\langle a, b\rangle \langle b, c\rangle \langle c, d\rangle \ldots \langle m, n\rangle]$, où chaque tuple de rang i est une instance de la propriété P_i, par rapport à l'interprétation I, $\forall i \in [1, n-1]$. L'élément a est appelé *origine* du $\rho - path$ et l'élément n *terminus.*

Pour identifier l'ensemble des nœuds qui peuvent être inclus dans une séquence de propriétés, la fonction $NodesOfPS()$ a été définie tel que $NodesOfPS(S) = \{C_1, C_2, \ldots, C_k\}$, tel que $\forall i \in [1..k], \exists j \in [1..n], C_i \in domaine(P_j)$ ou $C_i \in image(P_j), P_j \in S$. Afin, d'identifier les individus inclus dans un $\rho - path$ la fonction $PSNodesSequence()$ a été définie tel que $PSNodesSequence(\alpha) = \{a, b, \ldots, n, m\}$ [19].

Comme exemple, on peut considérer les séquences de propriétés $S_1 = \lfloor \texttt{crée} \bullet \texttt{exposé} \rfloor$, $S_2 = \lfloor \texttt{peint} \bullet \texttt{exposé} \rfloor$, $S_3 = \lfloor \texttt{sculpte} \rfloor$, $S_4 = \lfloor \texttt{peint} \bullet \texttt{matériel} \rfloor$, $S_5 = \lfloor \texttt{sculpte} \bullet \texttt{exposé} \rfloor$, construites pour le graphe RDF(S) de la figure 4.1. La séquence S_1 est interprétée comme étant un ensemble de trois $\rho - paths$: $I(S_1) = \{[\langle r_1, r_2\rangle, \langle r_2, r_3\rangle], [\langle r_4, r_5\rangle, \langle r_5, r_8\rangle], [\langle r_6, r_7\rangle, \langle r_7, r_8\rangle]\}$, illustrés de façon graphique dans la figure 4.2.

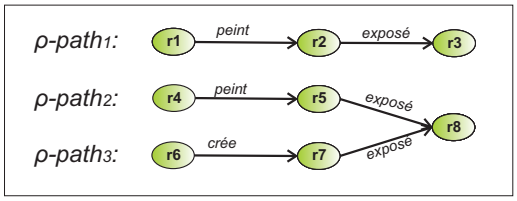


FIG. *4.2: Notation graphique pour les instances attachés à la séquence de propriétés* S_1.

En revanche, l'interprétation de la séquence S_4 est vide, car l'image de la propriété `peint` et le domaine de la propriété `matériel` n'ont pas d'objets communs, donc, dans l'état actuel de la ABox, il n'y a pas de possibilité de construire un $\rho - path$ pour la séquence S_4.

Anyawu et Sheth [19, 18] mettent en évidence deux types particuliers de séquences de propriétés : les séquences $\rho - jointes$ (voir définition 4.2) et les séquences $\rho - isomorphes$ (voir définition 4.3).

Définition 4.2. *Un ensemble de séquences de propriétés* $S_1, S_2, S_3, \ldots, S_n$ *sont* $\rho - jointes$ *si les deux conditions suivantes sont satisfaites :*

a) $S_1, S_2, S_3, \ldots, S_n$ *sont réalisables dans une même interprétation* I,

b) $\bigcap NodesOfPS(S_i) \neq \emptyset, \forall i = 1..n$.

En d'autres mots, plusieurs séquences de propriétés sont considérées comme jointes si elles passent toutes par une certaine classe "nœud" $C, C \in IC$. Dans l'ensemble de séquences de propriétés considérées auparavant les séquences S_1 et S_2 sont $\rho - jointes$, car elles partagent la classe nœud, `Musée` :

$$NodesOfPS(S_1) = \{\texttt{Artiste}, \texttt{Artefact}, \texttt{Musée}\},$$
$$NodesOfPS(S_2) = \{\texttt{Peintre}, \texttt{Peinture}, \texttt{Musée}\},$$
$$NodesOfPS(S_1) \cap NodesOfPS(S_2) = \{\texttt{Musée}\}.$$

Définition 4.3. *Deux séquences de propriétés* $S_1 = \lfloor P_1 \bullet P_2 \bullet \ldots \bullet P_m \rfloor$ *et* $S_2 = \lfloor Q_1 \bullet Q_2 \bullet \ldots \bullet Q_m \rfloor$ *définies pas rapport à un graphe RDF(S) de référence* R, *sont* $\rho - isomorphes$ *(*$S_1 \cong_\rho S_2$*) si et seulement si :*

a) S_1 *et* S_2 *sont réalisables par rapport à une interprétation* I, *de* R *et*

b) $\forall i \in 1..m$ *les deux conditions suivantes sont respectées :*
   - *(i)* $P_x = Q_x$ *ou* $P_x \sqsubseteq Q_x$ *ou* $Q_x \sqsubseteq P_x$, *(le symbole* $\sqsubseteq$ *représente la relation de subsomption entre propriétés),*
   - *(ii)* $\exists x, y, z \in IC : x = domaine(P_x), y = domaine(Q_x)$[11], $z = PlusPetitSubsumantCommun(x, y)$, *et les distances sémantiques entre* x *et* z $(distance(x,z))$ *et* y *et* z $(distance(y,z))$ *sont minimales.*

Cette définition garantit la similarité des propriétés de même index incluses dans S_1 et S_2, selon deux dimensions : dans un premier temps la définition impose une contrainte de similarité entre les propriétés elles-mêmes (condition *(i)*), et, dans un deuxième temps la contrainte de similarité est imposée pour les domaines des propriétés de même index (condition *(ii)*). Examinons par

[11] La fonction *IC* est définie dans la section 3.2.2.1.

exemple, les séquences de propriétés $S_1 = \lfloor \texttt{peint} \bullet \texttt{exposé} \rfloor$ et $S_2 = \lfloor \texttt{crée} \bullet \texttt{exposé} \rfloor$, définies par rapport au graphe RDF(S) illustré dans la figure 4.1. Parce que la propriété `peint` est une sous-propriété directe de la propriété `crée`, elles sont considérées comme similaires. Par rapport aux domaines des propriétés du même index, les auteurs de [19] proposent l'utilisation d'une fonction de distance pour éviter que la relation entre elles soit basée sur une classe parent trop éloignée dans la hiérarchie pour pouvoir témoigner d'une similarité forte. Le calcul de cette distance peut être adapté aux exigences de l'utilisateur. Pour l'exemple choisi, la classe `Peintre` étant une sous-classe directe de la classe `Artiste`, leur similarité est très forte. Ainsi, les deux séquences de propriétés, S_1 et S_2, sont $\rho - isomorphes$.

L'exemple choisi dans [19, 18] pour illustrer la définition des séquences $\rho - jointes$ et $\rho - isomorphes$ est quelque peu trompeur, car les séquences S_1 et S_2 sont $\rho - jointes$ et également $\rho - isomorphes$. Toutefois, il faut préciser que les deux notions sont tout à fait différentes, parce que les séquences de propriété $\rho - jointes$ ne sont pas tenues d'être similaires.

Définition 4.4. *La longueur d'une séquence de propriétés est égale au nombre de propriétés que cette séquence contient.*

Définition 4.5. *La longueur de deux séquences de propriétés $\rho - jointes$, S_1 et S_2, est calculée comme étant le nombre de propriétés entre l'origine de la séquence S_1 et la classe de jointure plus le nombre de propriétés entre la classe de jointure et l'origine de la deuxième séquence S_2, c'est-à-dire le nombre de propriétés contenues par le chemin qui relie les origines des deux séquences.*

Par exemple, pour le graphe RDF(S) de la figure 4.1, la séquence de propriétés $\rho - jointes$ $S_{1,2} = \lfloor \texttt{peint} \bullet \texttt{exposé}, \texttt{crée} \bullet \texttt{exposé} \rfloor$ a la longueur 4.

4.3.2 Types d'associations sémantiques

À partir de la notion de séquence de propriétés, Anyanwu et Sheth définissent trois types d'associations sémantiques [19, 18], illustrés dans la figure 4.3.

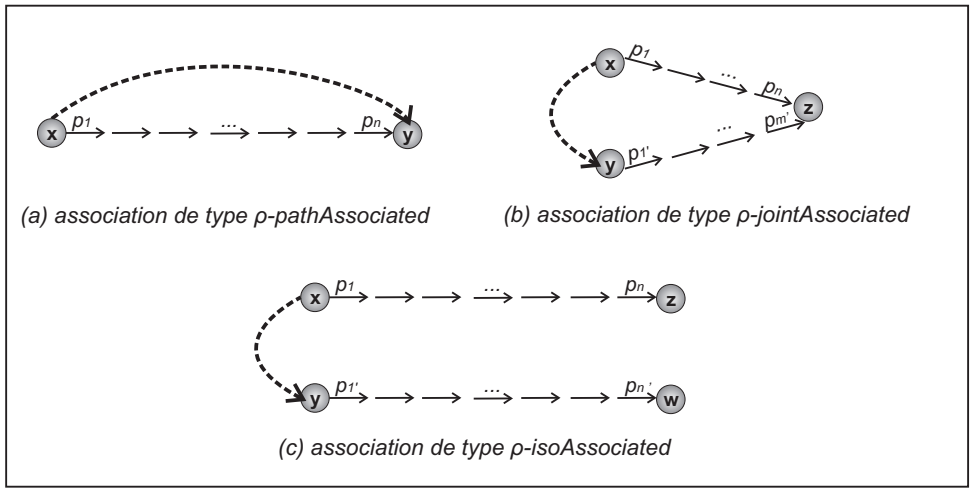


FIG. *4.3: Types d'association sémantiques.*

Définition 4.6. *Deux entités x et y, décrites dans un graphe RDF(S) de référence, R, ont une association sémantique de type $\rho - pathAssociated$ s'il existe au moins une séquence de propriétés SP qui est satisfaite dans une interprétation I de R, tel que $\exists ps \in I(PS)$ et soit x est*

l'origine et y le terminus de ps, soit l'inverse. Dans ce cas, on dit que ps satisfait la relation $\rho - pathAssociated(x,y)$ *:* $ps \models \rho - pathAssociated(x,y)$.

Si, entre deux individus *x* et *y*, il existe une relation de type $\rho - pathAssociated$, cela signifie que, dans le graphe RDF(S) de référence, il existe un chemin qui relie *x* à *y* directement, ou en passant par des individus intermédiaires (voir figure 4.3 (a)).

Définition 4.7. *Considérons deux séquences de propriétés* $\rho - jointes$, S_1 *et* S_2, *définies par rapport à un graphe RDF(S) de référence, R, et ayant comme nœud de jointure la classe* $C \in IC$. *S'il existe une interprétation I de R tel que* $\exists ps_1 \in I(S_1)$ *et* $\exists ps_2 \in I(S_2)$ *et* $\exists n \in ps1.PSNodesSequence() \cap ps2.PSNodesSequence()$, *alors une relation* $\rho - jointAssociated$ *existe entre x et y si l'une des conditions suivantes est satisfaite :*

a) *x est l'origine du* $\rho - path\ ps_1$, *et y est l'origine du* $\rho - path\ ps_2$ *ou*

b) *x est le terminus du* $\rho - path\ ps_1$, *et y est le terminus du* $\rho - path\ ps_2$.

Dans ce cas, on dit que les $\rho - paths\ (ps_1, ps_2)$ *satisfont la relation* $\rho - jointAssociated$ *:* $(ps_1, ps_2) \models \rho - jointAssociated(x,y)$.

Si, entre deux individus *x* et *y*, il existe une relation de type $\rho - jointAssociated$ (voir figure 4.3 (b)), cela signifie que, dans le graphe RDF(S) de référence, les individus *x* et *y* sont tous les deux associés à un troisième individu *z* par des relations de type $\rho - pathAssociated$. Par exemple, une association de type $\rho - jointAssociated$ existe entre deux collectionneurs intéressés par le même artiste ou ayant des œuvres du même artiste.

Définition 4.8. *Considérons deux séquences de propriétés* S_1 *et* S_2 *telles que* $S_1 \cong_\rho S_2$, *définies par rapport à un graphe RDF(S) de référence R. S'il existe une interprétation I de R qui satisfait les deux séquences de propriétés :* $\exists ps_1 \in I(S_1)$ *et* $\exists ps_1 \in I(S_2)$, *et en plus x est l'origine de* ps_1 *et y est l'origine de* ps_2, *alors les individus x et y sont reliés par une relation de type* $\rho - isoAssociated$. *Dans ce cas, on dit que la paire* (ps_1, ps_2) *satisfait la relation* $\rho - isoAssociated$ *:* $(ps1, ps2) \models \rho - isoAssociated\ (x,y)$.

Si, entre deux individus *x* et *y*, il existe une relation de type $\rho - isoAssociated$ (voir figure 4.3 (c)), cela signifie que, dans le graphe RDF(S) de référence, les individus *x* et *y* sont tous les deux associés à des individus du même type par des chemins similaires. Dans la pratique, ce type de relation est utilisé pour identifier des *motifs*. Par exemple, des individus ayant des parcours scolaires et/ou professionnels similaires, des types d'itinéraires préférés par les touristes, *etc.*

Définition 4.9. *Deux entités x et y, décrites dans un graphe RDF(S), R, ont une association sémantique par rapport à une interprétation I de R, si au moins une des relations suivantes est satisfaite :*

a) $\rho - pathAssociated\ (x,y)$

b) $\rho - jointAssociated\ (x,y)$

c) $\rho - isoAssociated\ (x,y)$.

En d'autres termes, lorsque on parle d'une *association sémantique* entre deux individus *x* et *y*, on peut faire référence à :

a) une association de type $\rho - pathAssociated$ qui relie *x* à *y* ou *y* à *x* (voir figure 4.3 (a)),

b) à deux $\rho - paths\ jointes$ dont l'origine est x respectivement y, témoins d'une association de type $\rho - jointAssociated$ (voir figure 4.3 (b)),

c) à deux $\rho - paths$ $\rho - isomorphes$ dont l'origine est x respectivement y, qui impliquent une association de type $\rho - isoAssociated$ (voir figure 4.3 (c)).

4.3.3 Requêtes sémantiques

Une requête sémantique est définie en utilisant l'opérateur binaire ρ, et a comme paramètres les individus x et y pour lesquels on désire mettre en évidence des associations sémantiques [19, 18]. Les résultats d'une requête sémantique représentent un ensemble de $\rho - paths\ p_i$ ou de paires de $\rho - paths\ (p_i, p_j)$ tel que :

a) $p_i \models \rho - pathAssociated(x,y)$ ou

b) $(p_i, p_j) \models \rho - isoAssociated(x,y) \mid \rho - jointAssociated(x,y)$.

4.3.4 Contexte d'une requête sémantique

Un autre élément très important de l'analyse sémantique est la définition et l'utilisation des *contextes de recherche* pour délimiter l'espace de recherche et pour classifier les résultats. La notion de contexte reduit la portée d'une requête utilisateur [219, 19, 124], et capture des informations sur les relations et les concepts qui sont particulièrement importants pour une requête donnée. Ceci permet de réduire de façon très substantielle l'espace de recherche et de classifier les résultats en fonction de leur pertinence pour le contexte donné (voir section 4.3.5).

Une *association sémantique* entre deux individus x et y, composée de plusieurs *tuples* qui relient des objets, peut passer par différentes *régions* du graphe RDF(S) de référence, R [124]. Une région désigne un ensemble de classes et de propriétés RDFS, définies par le graphe R. Halaschek *et al.* [124] utilisent la notion de *région* pour spécifier l'intérêt de l'utilisateur par rapport à une requête sémantique donnée. Par exemple, pour le graphe RDF(S) illustré dans la figure 4.1, et pour la requête sémantique $\rho(r_4, r_8)$, l'utilisateur peut spécifier plusieurs régions d'intérêt telles que : $R_1 = \{\texttt{Peintre}, \texttt{Peinture}, \texttt{peint}, \texttt{Musée}\}$ ou $R_2 = \{\texttt{Artiste}, \texttt{crée}\}$. La première *région*, R_1, indique que l'utilisateur est intéressé par un domaine très spécifique, celui de la peinture, dans lequel les acteurs principaux sont les `Peintres`, les `Peintures` et les `Musées`. La relation qui compte le plus pour l'utilisateur est `peint` définie entre un `Peintre` et sa `Peinture`. La deuxième région désigne un intérêt plus large, celui pour les `Artistes` en général et pour leurs créations (`crée`).

Les *régions* d'intérêt ne sont pas nécessairement disjointes. Pour chaque *région* d'intérêt, l'utilisateur peut spécifier un poids ($w \in [0,1]$) qui quantifie l'importance qu'il donne à l'ensemble correspondant de classes et de propriétés. Le poids associé à une *région* d'intérêt est également attaché à toutes les instances de classes et de propriétés contenues dans la *région* en question. Ainsi, chaque instance (objet ou *tuple*) contenue par le graphe RDF(S) interrogé, peut avoir zéro ou plusieurs poids associés, en fonction du nombre de contextes dans lesquelles sa classe/relation est incluse. Ce poids est utilisé lors de l'étape de classification des résultats d'une requête sémantique.

L'ensemble des *régions* d'intérêt définies par l'utilisateur pour une requête sémantique donnée constitue le *contexte thématique*, C_t, de la requête. Dans l'exemple précédent, le contexte thématique est $C_t = \{R_1, R_2\}$.

4.3.5 Classification des résultats

Il est manifeste que le nombre de chemins sémantiques liant deux individus, découverts à travers l'*analyse sémantique* lors de l'interrogation de plusieurs graphes RDF(S) inter-reliés et ayant un grand nombre d'individus, ou même lors de l'interrogation du Web Sémantique, peut se révéler trop grand pour que l'utilisateur, ou même les agents logiciels, puissent les exploiter de façon efficace. Les raisons pour cela sont : *i)* l'étendue très large de la base de connaissances à exploiter (*i.e.* le Web Sémantique, qui peut contenir des millions d'assertions rassemblées dans des graphes RDF(S) inter-reliés), et *ii)* le fait qu'on ne s'intéresse pas seulement aux relations explicitement stockées dans des graphes RDF(S), mais aussi à celles inférées et aux associations de type $\rho - pathAssociated$, $\rho - isoAssociated$ et $\rho - jointAssociated$, de longueur variable, qui existent entre x et y. Il est donc nécessaire de déterminer un classement de ces relations, sur la base de leur pertinence par rapport au contexte de la requête.

Pour choisir parmi ces résultats les *associations sémantiques* pertinentes, Aleman-Meza *et al.* [10, 11] proposent plusieurs critères de classement, chacun correspondant à une formule mathématique pour quantifier le poids d'une *association sémantique*. Les auteurs proposent de prendre en compte le contexte thématique (ensemble des concepts et des relations d'une ontologie qui présentent un intérêt pour l'utilisateur), la longueur du chemin (le nombre d'entités et de relations contenues par une association sémantique), ainsi que la confiance dans la source de données. Néanmoins, les travaux que nous avons étudiés n'adaptent pas le calcul de la pertinence des résultats d'une requête sémantique aux associations sémantiques de type $\rho - isoAssociated$ ou de type $\rho - jointAssociated$, mais traitent exclusivement les associations de type $\rho - pathAssociated$.

Le contexte thématique

Le critère de classification le plus important est le *contexte thématique*, C_t (la définition du contexte thématique est donnée dans la section 4.3.4). Un poids contextuel, W_{C_t}, est calculé pour chaque *association sémantique* α obtenue comme résultat d'une requête sémantique donnée, $\rho(x,y)$, en fonction des poids attachés aux éléments auxquels α fait référence.

Quelques précisions sont nécessaires avant de pouvoir définir formellement ce critère de classification. Considérons α, une *association sémantique* de type $\rho - pathAssociated$ (voir définition 4.6), définie comme un ensemble de *tuples* : $\alpha = [\langle x, b_1\rangle, \langle a_2, b_2\rangle, \ldots \langle a_n, y\rangle]$. Exceptés les *tuples* (instances de propriétés) $\langle a_k, b_k\rangle, k \in [1,n]$,(où $a_1 = x$ et $b_n = y$), α fait des références implicites aux objets (instances de classes) reliés par ces *tuples* $:x, b_1, a_2, \ldots, a_n, y$. Chaque instance (*tuple* ou objet) référencée par α peut avoir zéro ou plusieurs poids associés, en fonction du nombre de contextes dans lesquelles sa classe/relation est incluse. Pour calculer le poids contextuel W_{C_t} attaché à un $\rho - path$ α, Aleman-Meza et al. [10] retiennent, pour chaque objet et *tuple* contenu par α, seulement le plus grand des poids associés aux *régions* d'intérêt dont il fait partie.

Considérons également le contexte thématique $C_t = \{R_1, R_2, \ldots, R_k\}$, où chaque région d'intérêt R_i a un poids W_{R_i} attaché. Pour des raisons de commodité, plusieurs ensembles intermédiaires sont définis dans [11] qui regroupent les entités référencées par α (objets et *tuples*) dont le poids est supérieur à zéro ($X_i(\alpha), i \in [1..k]$), ou égal à zéro ($Z(\alpha)$) :

$$X_i(\alpha) = \{c \mid c \in \alpha \wedge c \in R_i\}, \forall i \in [1..k] \text{ [12]}$$
$$Z(\alpha) = \{c \mid c \in \alpha \wedge \nexists i \in [1..k], c \in R_i\}.$$

[12] Ici, la notation $c \in R_i$ représente le fait que le type de c est inclus dans la région R_i.

Ainsi, les ensembles $X_i(\alpha)$ regroupent les éléments référencés par α dont les types (classe ou propriété) sont inclus dans les régions d'intérêt R_i, tandis que l'ensemble $Z(\alpha)$ regroupe les éléments de α qui ne présentent pas d'intérêt pour l'utilisateur.

Il est important de préciser que les calculs liés au calcul du poids d'une *association sémantique*, notamment la construction des ensembles X_i et Z, ainsi que le calcul de la longueur totale d'un $\rho - path$, ne tiennent pas compte de l'*origine* et du *terminus* du $\rho - path$ car, pour une requête sémantique donnée, les deux éléments sont les mêmes dans toutes les $\rho - paths$ résultats.

En guise d'illustration, nous considérons l'exemple du graphe RDF(S) de la figure 4.4. Il a été développé par le laboratoire *LSDIS* pour décrire un domaine relié à la sécurité d'un territoire [12]. Dans ce graphe, deux *régions* d'intérêt ont été définies. La première, R_1, fait référence à des concepts et événements liés au terrorisme,

$$R_1 = \{\texttt{TerroristOrganization}, \texttt{TerroristAttackEvent}, \texttt{SuicideAttack}, \texttt{TerroristTarget}, \texttt{doesBusinessWith}, \texttt{involvedIn}, \texttt{where}\}$$

et a le poids le plus élevé, $W_{R_1} = 0.75$, tandis que la deuxième *région* d'intérêt, R_2, fait référence à des organisations financières et aux relations d'affaires et d'appartenance à des organismes financiers :

$$R_2 = \{\texttt{FinancialOrganization}, \texttt{BusinessOrganization}, \texttt{memberOf}, \texttt{locatedIn}\}$$

et a un poids moins important : $W_{R_2} = 0.5$. À partir de ces poids, on peut déduire que l'utilisateur est intéressé principalement par le domaine du terrorisme, mais souhaite également prendre en compte, avec une priorité inférieure, les associations ayant trait à la finance.

La figure 4.5 illustre trois exemples de réponses à la requête sémantique $\rho(e_1, e_9)$, qui passent par les deux *régions* d'intérêt, R_1 et R_2. Le premier $\rho - path$, $\alpha_1 = \lfloor \langle e_1, e_2 \rangle, \langle e_2, e_3 \rangle, \langle e_3, e_4 \rangle, \langle e_4, e_9 \rangle \rfloor$, fait référence à deux instances incluses dans la région d'intérêt R_2 : $e_2, e_6 \in R_2$ ainsi qu'à une instance incluse dans la région R_1 : $e_4 \in R_1$. Le deuxième $\rho - path$, $\alpha_2 = \lfloor \langle e_1, e_5 \rangle, \langle e_5, e_6 \rangle, \langle e_6, e_9 \rangle \rfloor$, inclut un seul élément qui fait partie d'une région d'intérêt, l'instance $e_6 \in R_2$. Le troisième $\rho - path$, $\alpha_3 = \lfloor \langle e_1, e_7 \rangle, \langle e_7, e_8 \rangle, \langle e_8, e_9 \rangle \rfloor$ contient exclusivement des objets inclus dans la région R_1. De façon intuitive, en regardant le contexte de la requête et les trois résultats, on se rend compte que le $\rho - path$ α_3 est le plus pertinent et devrait être classé en premier. Les objets qu'il relie sont tous inclus dans la région d'intérêt la plus importante pour l'utilisateur.

En suivant ces intuitions, Aleman-Meza *et al.* [12] définissent le poids contextuel d'une association α en utilisant l'équation 4.1 :

$$W_{C_t}(\alpha) = \frac{1}{2n-1} \left(\sum_{i=1}^{k} (W_{R_i} \times |X_i(\alpha)|) \right) \left(1 - \frac{|Z(\alpha)|}{2n-1} \right) \tag{4.1}$$

Pour chaque *région* d'intérêt R_i qui contient des instances référencés par le $\rho - path$ analysé, α, on calcule le nombre d'éléments partagés entre R_i et α (nombre désigné par $X_i(\alpha)$) et on le multiplie par le poids W_{R_i} attaché à R_i. Pour récompenser les $\rho - paths$ dont tous les composants font partie du contexte, on utilise le rapport entre les éléments n'appartenant à aucun contexte ($Z(\alpha)$)

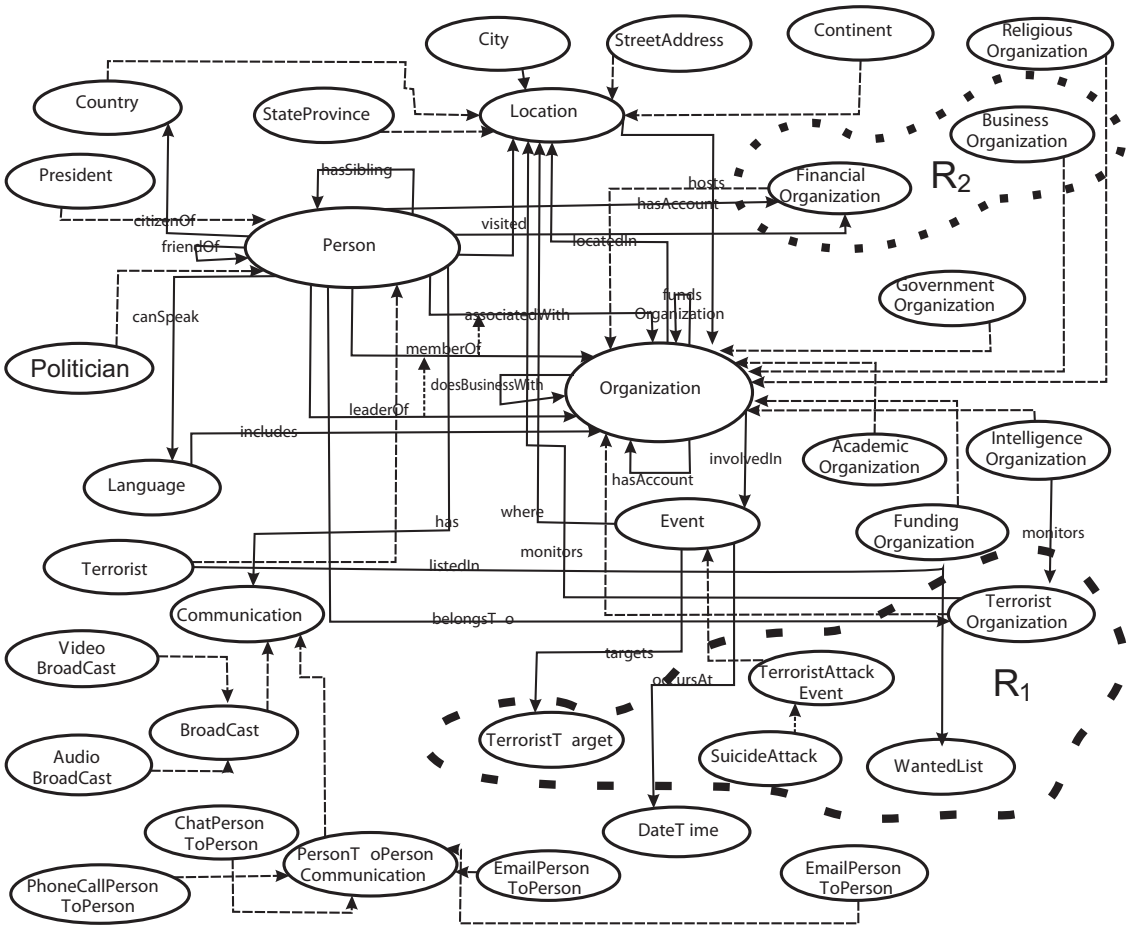


FIG. *4.4: Exemple de graphe RDF(S) complexe dans le domaine de la sécurité d'un territoire (source [12])*

.

et la *longueur totale*[13] de l'association sémantique (2n-1), qu'on soustrait de 1 (voir équation 4.1). Le résultat est multiplié par la somme précédente et normalisé par la *longueur totale* du $\rho - path$.

Afin d'illustrer le calcul du poids attaché à un $\rho - path$, nous considérons les exemples présentés dans la figure 4.5. Pour le $\rho - path$ α_1 on obtient la somme de départ $(0.75 \times 3) + (0.5 \times 3) = 3.75$. Un seul élément du $\rho - path$ α_1 n'est pas attaché à une classe/propriété incluse dans une région d'intérêt : e_3, donc on obtient : $3.75 \times \left(1 - \frac{1}{7}\right) = 3.21$, somme normalisée par le nombre total d'éléments de α_1 : $W_{C_t}(\alpha_1) = \frac{1}{7} \times 3.21 = 0.458$.

Le $\rho - path$ α_2 passe exclusivement par la région R_2. Donc, on obtient la somme initiale $(0.5 \times 3) = 1.5$. Il y a deux éléments de α_2 qui ne sont inclus dans aucune région, donc on pénalise α_2 de $1 - \frac{2}{5}$: $1.5 \times \left(1 - \frac{2}{5}\right) = 0.9$. Ce résultat est normalisé par la longueur totale de α_2. Le résultat final est $W_{C_t}(\alpha_2) = \frac{1}{5} \times 0.9 = 0.18$, qui représente la pertinence de α_2 par rapport au contexte de la requête. En suivant la même démarche, on obtient le poids de α_3 comme étant, $W_{C_t}(\alpha_3) = 0.75$. Cette méthode de classement attache le plus de pertinence au résultat α_3, puis α_1, et finalement α_2.

[13] Cette *longueur totale* est calculée comme le nombre de *tuples* inclus par l'association sémantique plus le nombre d'objets référencés de façon implicite. L'origine et le terminus du $\rho - path$ sont ignorés, car ils ne changent pas pour une requête donnée.

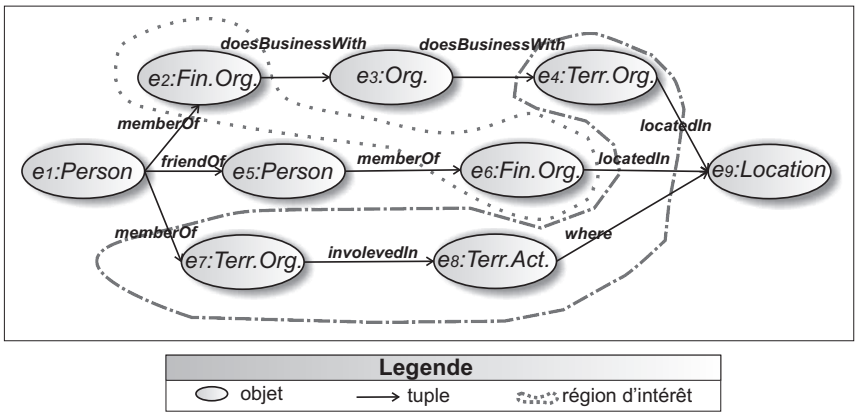


FIG. *4.5: Exemple d'associations sémantiques qui passent par plusieurs régions d'intérêt.*

La longueur de l'*association sémantique*

Le deuxième critère de classement repose sur la longueur de l'*association sémantique* étudiée. Selon que l'utilisateur est intéressé par des chemins plus directs (plus courts) qui témoignent généralement d'un lien fort, ou par de longs chemins modélisant des liens cachés ou indirects, Aleman-Meza *et al.* [12] proposent deux modalités de calcul de la pertinence d'un $\rho - path$:

$$W_L(\alpha) = \frac{1}{2n-1} \ (a); \quad W_L(\alpha) = 1 - \frac{1}{2n-1} \ (b); \tag{4.2}$$

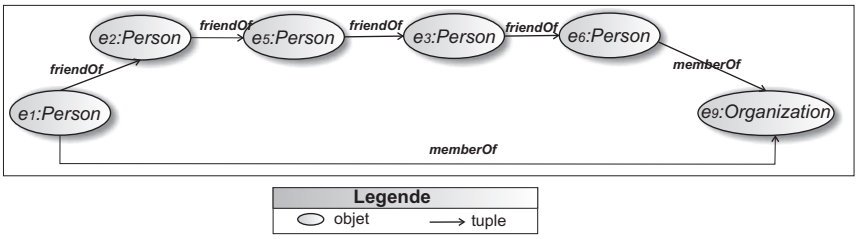


FIG. *4.6: Exemple de $\rho - paths$ de longueur differentes.*

La deuxième équation, 4.2 (b), est plus utile dans des domaines où peuvent exister des tentatives délibérées pour masquer les relations entre individus. Par exemple, les cellules terroristes potentielles vont rester éloignées en évitant le contact direct afin de contourner les mesures de détection. C'est également le cas pour le blanchiment de capitaux lorsque des chaînes de transactions financières cherchent délibérément à cacher la provenance de l'argent. Ainsi, la méthode utilisée pour le calcul du poids W_L est influencée par le domaine.

Si on considère les deux $\rho - paths$ de la figure 4.6 qui relient la personne e_1 et l'organisation e_9, dans l'hypothèse où l'utilisateur préfère les associations directes, alors le $\rho - path$ $\alpha_4 = \lfloor \langle e_1, e_2 \rangle, \langle e_2, e_5 \rangle, \langle e_5, e_3 \rangle, \langle e_3, e_6 \rangle, \langle e_6, e_9 \rangle \rfloor$, doit apparaître dans la liste des résultats après l'association $\alpha_5 = \lfloor \langle e_1, e_9 \rangle \rfloor$. En utilisant l'équation 4.2 (a) pour les deux associations sémantiques, on obtient $W_L(\alpha_4) = \frac{1}{9}$ tandis que $W_L(\alpha_5) = \frac{1}{1} = 1$. Ainsi, l'association la plus courte a un poids plus grand.

La confiance dans la source de données

Le troisième critère vise le niveau de confiance que l'utilisateur a vis-à-vis des assertions provenant de différentes sources. Un niveau de confiance $t \in [0,1]$ est affecté à chaque source. Chaque *tuple* se voit attribuer le niveau de confiance de sa source, ou le niveau par défaut qui est fixé à 1 [12]. Le niveau total de confiance d'un $\rho - path$ sera le produit des niveaux de confiance t de chaque instance de propriété qu'il inclut.

$$W_T(\alpha) = \prod_{i=1}^{n} t_{p_i}. \tag{4.3}$$

Une approche différente pour le calcul de la confiance attachée à une *association sémantique* est présentée dans [11]. Les auteurs suivent l'intuition selon laquelle la confiance maximale qu'on peut avoir en une *association sémantique* ne peut pas être supérieure à la confiance minimale qu'on a dans les *tuples* qu'elle contient (voir équation 4.4). Ils définissent la confiance d'un $\rho - path$ comme étant la confiance minimale des *tuples* inclus dans le $\rho - path$:

$$W_T(\alpha) = min(t_{p_i}), i \in [1,n]. \tag{4.4}$$

Le poids final attaché à une association sémantique sera calculé comme une combinaison pondérée des critères de classification présentés dans cette section (voir équation 4.5). Les facteurs $\varepsilon_i \in (0,1], \sum_{i=1}^{3} \varepsilon_i = 1$, permettent d'adapter la prise en compte (l'importance) des filtres aux préférences des utilisateurs.

$$W(\alpha) = \varepsilon_1 \times W_{C_t}(\alpha) + \varepsilon_2 \times W_L(\alpha) + \varepsilon_3 \times W_T(\alpha). \tag{4.5}$$

4.3.6 Limitations

Le choix d'utiliser RDF(S) comme langage de modélisation d'ontologies peut être considéré comme trop restrictif, compte tenu de son expressivité limitée par rapport aux logiques de description ou aux langages de représentation de connaissances par objets. Parmi les limitations les plus importantes de RDF(S) au regard de l'analyse sémantique, on peut citer l'absence d'axiomes pour la définition d'alignement de graphes RDF(S), ainsi que le nombre limité d'axiomes mis à disposition par le langage (seuls `subClassOf` et `subPropertyOf`) .

Également, l'importation d'un graphe RDF(S), *S*, dans un autre graphe RDF(S), *D*, correspond en fait à la mise en commun des deux ensembles de triplets contenus par les deux graphes. Les possibilités d'appariement sont alors très limitées et basées exclusivement sur les relations de type `sous-classe` et `sous-propriété`. Or, il n'est pas possible, par exemple, de spécifier deux instances comme étant égales, ou deux classes ou deux propriétés comme étant équivalentes. Donc, même réunis, les deux ensembles de triplets *S* et *D* restent "*isolés*". Cela peut empêcher la découverte de $\rho - paths$ d'une grande pertinence.

Étudions, par exemple, la requête sémantique $\rho(pers_1, pers_4)$ pour les graphes RDF(S) présentés sur la figure 4.7, en sachant que l'individu $pers_2$, défini dans le graphe `graph1.rdf`, est le même que l'individu $pers_3$, défini par le graphe `graph2.rdf`. En RDF(S), les graphes `graph1.rdf` et `graph2.rdf` restent déconnectés car la définition d'appariement n'est pas possible. Dans cette situation, il est impossible de trouver une association sémantique entre $pers_1$ et $pers_4$.

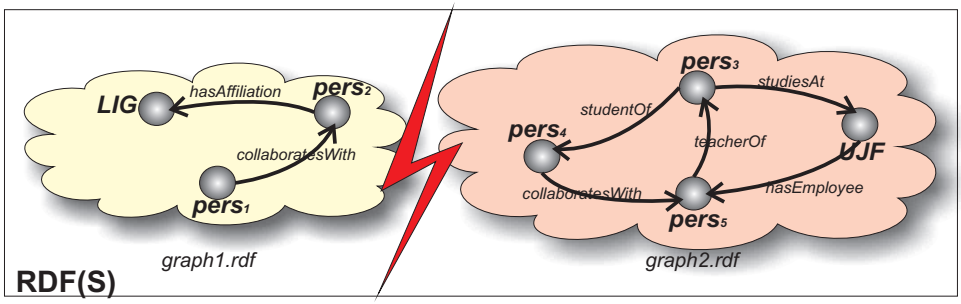


FIG. *4.7: Illustration des conséquences liées à l'impossibilité d'établir des appariements entre graphes RDF(S).*

Ces problèmes ont été dépassés par des langages plus évolués comme OWL ou OWL 2 qui offrent des primitives dédiées à la définition des *alignements d'ontologies*. De plus, des travaux récents [91, 79, 240] proposent des outils automatiques pour l'alignement d'ontologies, qui peuvent être utilisés pour déterminer les équivalences existant entre différentes ontologies, et qui seront ensuite exploitées pour la déduction d'associations sémantiques.

Ainsi, si les informations contenues par les graphes `graph1.rdf` et `graph2.rdf` sont modélisées en OWL-DL, ou OWL 2 DL, la définition d'une relation d'égalité entre $pers_2$ et $pers_3$ est possible (voir figure 4.8). Celle-ci crée un pont entre les deux ontologies `onto1.owl` et `onto2.owl`, ce qui rend possible la découverte d'au moins une relation sémantique :

$p_1 = \lfloor \langle pers_1, pers_2 \rangle, \langle pers_2, pers_4 \rangle \rfloor$.

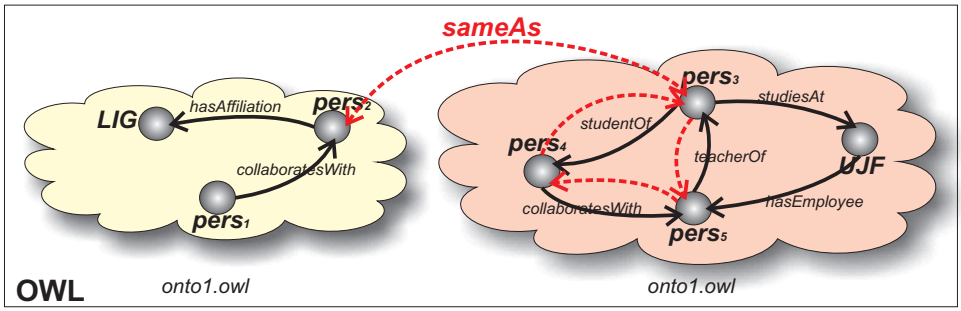


FIG. *4.8: Illustration de l'appariement possible en OWL qui permet la découverte d'une relation sémantique.*

L'ensemble d'axiomes mis à disposition par RDF(S) est très limité. Ainsi, les inférences qu'on peut réaliser à partir d'un ensemble de triplets sont réduites. Ces limitations sont propagées au processus de découverte des associations sémantiques qui va exploiter exclusivement les connaissances explicitement stockées dans les graphes RDF(S) de référence.

En revanche, OWL-DL et OWL 2 DL offrent un ensemble étendu d'axiomes qui permettent de définir, par exemple, deux propriétés comme étant inverses, fonctionnelles, symétriques, *etc.* Pour l'exemple précédent, si on définit les propriétés `studentOf` et `teacherOf` comme étant inverses, les tuples implicites (symbolisés par des flèches en pointillés dans la figure 4.8) `teacherOf`$(pers_4, pers_3)$ et `studentOf`$(pers_3, pers_5)$ peuvent être déduits en utilisant un raisonneur tel que Pellet ou RacerPro. Si on définit également la relation `collaboratesWith` comme étant symétrique, deux nouvelles association sémantiques reliant les individus $pers_1$ et $pers_4$ peuvent être découvertes :

$p_2 = \lfloor \langle pers_1, pers_2 \rangle, \langle pers_2, pers_5 \rangle, \langle pers_5, pers_4 \rangle \rfloor$

$p_3 = \lfloor \langle pers_1, pers_2 \rangle, \langle pers_2, UJF \rangle, \langle UJF, pers_5 \rangle, \langle pers_5, pers_4 \rangle \rfloor$.

4.4 Associations sémantiques géospatiales

Jusqu'ici, l'analyse sémantique s'est essentiellement centrée sur la dimension thématique des métadonnées, en analysant, par exemple, les liens de collaboration entre deux personnes, membres d'une organisation. Néanmoins, étant donnée la popularité croissante des applications spatiales sur le Web (Google Earth, Mappy, ViaMichelin, Geoportail, Virtual Earth 3D, *etc.*), plusieurs travaux récents promeuvent la prise en compte des dimensions spatiales et temporelles des données lors de l'analyse sémantique en tant qu'un enjeu crucial [123, 200, 199].

Cette gestion contribuera à la fois à déduire davantage d'associations sémantiques, et à rejeter celles présentant des incompatibilités avec le contexte spatio-temporel de la requête. Par exemple, partant d'une description ontologique de deux personnes qui ont vécu dans la même résidence pendant dix ans, une analyse sémantique spatiale et temporelle pourra déduire qu'il est très probable que les deux personnes se connaissent. La proximité et la simultanéité de résidence est ici automatiquement prise en compte par l'analyse sémantique spatio-temporelle pour suggérer qu'il pourrait y avoir un lien entre ces deux personnes.

Afin d'étendre l'analyse sémantique vers la prise en compte des dimensions temporelle et spatiale, Hakimpour *et al.* [123] s'appuient sur l'utilisation de plusieurs schémas RDF génériques modélisant la spatialité et la temporalité des ressources. Ceux-ci ne sont pas des schémas standardisées, et permettent exclusivement la définition des descriptions spatiales et temporelles basiques (paires de coordonnées, adresses, instants et intervalles). Les seules relations qualitatives prises en compte sont les relations topologiques.

Pour les ressources décrites à l'aide des concepts spatio-temporels, ces auteurs définissent plusieurs opérateurs de filtrage : `associatedEvent`(type, resource, n), `nearestEvent`(type, pos, n), `nearestEventBefore`(type, t, n), `nearestEventAfter`(type, t, n). Ceux-ci sélectionnent les individus du type spécifié par l'utilisateur qui satisfont une condition de proximité spatiale ou temporelle vis-à-vis d'une région *pos* ou d'un instant t. Pour vérifier les relations de proximité, des distances euclidiennes simples sont calculées à partir des connaissances quantitatives explicitement stockées dans la base ontologique interrogée.

Une approche similaire, présentée par Perry *et al.* [200], propose l'intégration d'ontologies RDF et de bases de données ORACLE, à travers l'utilisation d'*ORACLE Semantic Data Store*, qui fournit les capacités de stocker, d'inférer, et d'interroger des données sémantiques sous la forme de simples descriptions RDF ou d'ontologies basées sur RDFS. La spatialité des ressources ontologiques, modélisée en RDF à l'aide de listes de coordonnées (par exemple, des instances de la classe Line), est traduite vers le type spatial *SDO_GEOMETRY*, afin de bénéficier des fonctionnalités d'interrogation proposées par ORACLE Spatial, notamment l'ensemble d'opérateurs spatiaux topologiques et de distance. Cette approche, outre le fait qu'elle soit basée sur l'utilisation d'un outil propriétaire, ne permet pas de combiner les raisonnements qualitatif et quantitatif. De plus, les informations spatiales et temporelles sont exclusivement utilisées pour filtrer les $\rho - paths$ en fonction des relations spatiales/temporelles vérifiées par leurs éléments vis-à-vis d'une région/intervalle de temps de référence donnée.

4.5 Synthèse

Après la présentation, dans le chapitre 3, du Web Sémantique et du Web Sémantique Géospatial, dans ce chapitre nous avons étudié les techniques d'exploitation des connaissances contenues par les pages Web, et notamment les techniques d'exploitation des annotations RDF(S).

Nous nous sommes concentrés principalement sur l'analyse sémantique, une nouvelle approche de découverte des relations complexes liant les individus décrits dans des graphes RDF(S). Nous avons présenté les différents types d'associations qui peuvent exister entre individus, ainsi que les méthodes utilisées pour leur classification. Pour limiter l'espace de recherche, l'analyse sémantique utilise la notion de contexte d'une requête.

Nous avons également présenté les principales limites de l'analyse sémantique, induites par l'utilisation du langage RDF(S), notamment l'impossibilité de définir des appariements entre les individus, les classes et les propriétés définies dans des graphes RDF(S) différents, ainsi que la palette réduite des axiomes proposés par le langage.

Nous avons enfin étudié deux travaux qui ouvrent l'analyse sémantique vers le Web Sémantique Géospatial. Ceux-ci utilisent les informations spatiales et temporelles en vue de filtrer les associations sémantiques par rapport à un contexte spatial et temporel donnés.

Deuxième partie

PROPOSITION

Comme souligné dans l'état de l'art, le contexte général de notre étude est le Web Sémantique Géospatial. Celui-ci peut être vu comme une extension du Web Sémantique (voir figure 4.9) qui garantit une gestion adaptée des annotations spatiales et temporelles quantitatives et qualitatives. Cette gestion est définie en termes de représentation et de raisonnement.

Dans cette thèse, nous défendons l'idée que la construction du Web Sémantique Géospatial ne peut que s'inspirer des domaines spécialisés dans la représentation et la gestion de l'espace et du temps (voir figure 4.9). Un de ces domaines est celui des Systèmes d'Information Géographique, dont les recherches sont centrées sur la définition de types et d'opérateurs spatiaux et temporels, ainsi que sur la gestion efficace de données spatiales/temporelles quantitatives. De plus, un autre défi à relever dans la concrétisation d'un Web Sémantique que l'on souhaiterait (aussi) Géospatial, réside dans la capacité à gérer des requêtes - et donc des données - flexibles et imprécises. Une formulation plus souple (car plus proche des termes employés dans le langage usuel) des requêtes adressées aux sources disponibles sur le Web peut être favorisée par l'usage de formalismes qualitatifs de représentation et de raisonnement (voir figure 4.9).

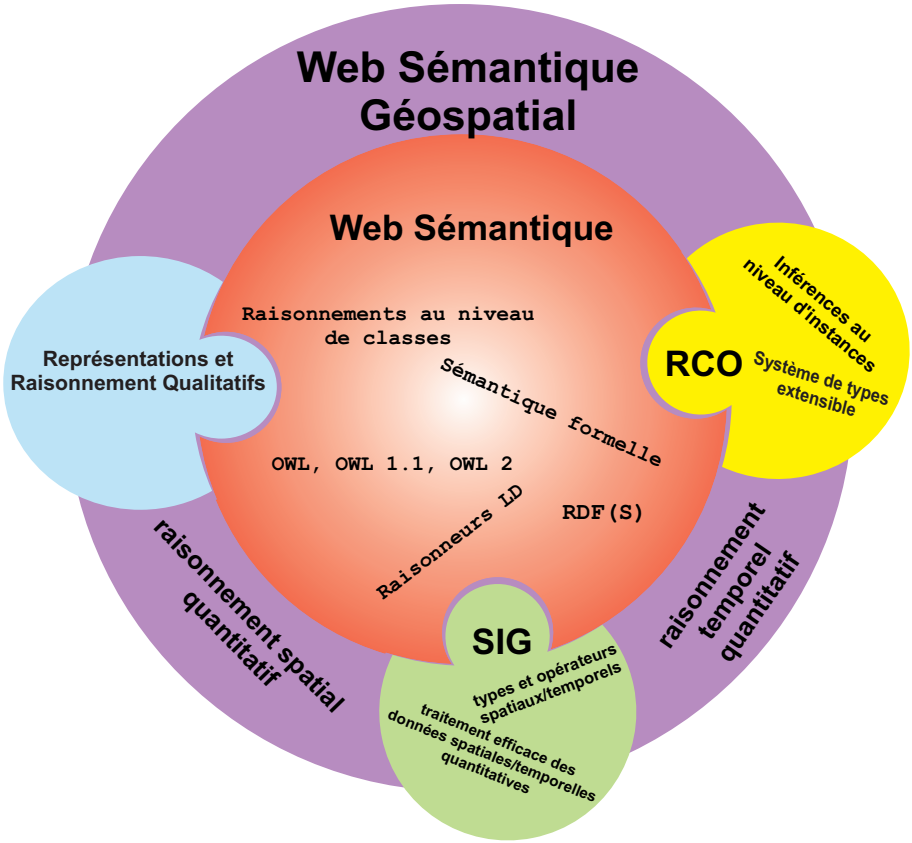


FIG. *4.9: Construction du Web Sémantique Géospatial*

Nous avons choisi d'utiliser le système de Représentation de Connaissances par Objets, appelé AROM, comme passerelle entre les Logiques de Description, qui sont à la base des langages d'ontologies, d'une part, et les SIG et le domaine du qualitatif, d'autre part (voir figure 4.9). Notre choix est basé sur :

– l'extensibilité du système AROM, qui a été déjà exploitée pour la définition d'un ensemble de types spatiaux et temporels [174],

– l'existence d'un Langage de Modélisation Algébrique, lui aussi facile à compléter par des opérateurs spatiaux et temporels,
– une étude qui montre les similarités en termes de représentation qui existent entre AROM et OWL DL [168].

Ainsi, le premier chapitre de la proposition (chapitre 5) illustre la construction d'un Système de Représentation de Connaissances par Objets, AROM-ONTO, comme extension du système AROM [193]. Le but de cette extension est d'obtenir un système de représentation et de raisonnement, capable de définir des bases de connaissance d'une expressivité équivalente à celle autorisée par les ontologies OWL 2 DL, et capable d'exploiter ces bases à travers des raisonnements complémentaires aux raisonnements disponibles pour les ontologies, notamment des raisonnements spatiaux et temporels.

Le deuxième chapitre de notre proposition (chapitre 6), présente la définition, au-dessus du modèle AROM-ONTO, d'un modèle spatial et temporel qualitatif. Ce modèle qualitatif est complété par un raisonneur spatial et temporel, capable d'exploiter de façon adaptée les connaissances spatiales et temporelles quantitatives et qualitatives modélisées à l'aide de types spatiaux et temporels, ou à l'aide de relations qualitatives. Cette nouvelle extension de AROM, appelée ONTOAST (acronyme de ONTOlogies en AROM ST), est capable de manipuler (importer, interroger, raisonner avec, sauvegarder) des ontologies OWL 2 DL, grâce au fait qu'elle est construite au-dessus du système AROM-ONTO.

Dans le chapitre 7, nous illustrons l'utilisation du système ONTOAST pour la découverte d'associations sémantiques entre individus, sur le Web Sémantique Géospatial. Pour cela, dans un premier temps, nous proposons l'adaptation de l'*analyse sémantique* pour les ontologies OWL 2 DL. Ensuite, nous définissons des algorithmes pour la découverte d'associations sémantiques, qui prennent en compte le contexte thématique, spatial et temporel de la requête. Nous utilisons le raisonneur ONTOAST pour vérifier la cohérence entre les caractéristiques spatiales et temporelles des individus et des tuples étudiés, et les contextes spatial et temporel. Les capacités de raisonnement de ONTOAST sont également utilisées pour la déduction des relations spatiales et temporelles implicites qui relient les individus et qui peuvent être utilisées pour la découverte des nouvelles associations sémantiques.

Chapitre 5

Un système hybride entre RCO et LD pour raisonner sur le Web Sémantique

À partir de prémisses fausses, tout peut se démontrer. Ce n'est pas parce qu'un raisonnement est implacable qu'il n'est pas délirant.

François Lelord

SOMMAIRE

Résumé

L'étude décrite dans ce chapitre positionne le Système de Représentation des Connaissances par Objets AROM par rapport au langage OWL 2 DL, du point de vue des capacités de représentation, de typage, et d'inférences. Nous mettons en évidence des points communs et des divergences. Nous soulignons un déficit d'AROM vis-à-vis de OWL 2 DL en termes de puissance de description, et, dans certains cas, nous proposons des solutions pour rapprocher le pouvoir expressif d'AROM de celui de OWL 2 DL.

5.1 Introduction

Partant du constat des faiblesses des langages d'ontologies (RDF(S), OWL, OWL 2...) en termes de représentation et de raisonnement spatiaux et temporels (voir chapitre 3), nous visons la proposition de solutions de modélisation et d'inférence sur lesquelles appuyer la construction du Web Sémantique Géospatial.

Pour cela nous avons étudié, dans un premier temps, les avantages et les inconvénients du modèle relationnel et de l'approche objet, en termes d'expressivité de la modélisation et de raisonnements mis à disposition, par rapport aux langages du Web Sémantique.

Les avantages des bases de données relationnelles s'expriment surtout en termes de performances et d'intégrité des données. Elles possèdent des systèmes évolués de sauvegarde et de vérification de cohérences. Ces bases sont prévues pour de systèmes d'information complexes nécessitant de hauts rendements. Néanmoins, le modèle relationnel est un modèle centralisé, qui impose une structure rigide, difficile à faire évoluer et dépendante des données elles-mêmes [228]. Il ne gère pas nativement l'interopérabilité entre différentes bases de données réparties sur un réseau, car il manque de sémantique formelle pour définir de façon générique le modèle de données utilisé [228].

Le succès du paradigme objet en représentation de connaissances repose sans doute sur la correspondance immédiate et directe qui peut être établie entre un modèle objet général, et une approche ontologique de la réalité. À proprement parler, le paradigme objet propose d'organiser et de gérer les connaissances autour de la notion d'*objet*, et de fournir des mécanismes d'inférence destinés à compléter l'information disponible. Les opérations principales qui concernent l'inférence de valeurs et le raisonnement sont le calcul de la valeur d'un attribut (procédure, filtrage, défaut), le test de subsomption, la classification d'instances. L'un des plus grands avantages des objets vient du fait qu'il est possible d'aligner la description formelle d'un Système de Représentation de Connaissances par des Objets (SRCO) sur celle d'une Logique de Description (LD) : un SRCO peut se voir comme un système logique qui fournit un langage de description des classes et des individus, et des procédures d'inférences qui reposent sur la subsomption [82].

Comme représentant du paradigme objet, nous avons choisi AROM [193] (décrit dans la section 5.2), un système de Représentation des Connaissances par Objets, conçu par l'action Romans de l'INRIA Rhône-Alpes et maintenu et développé depuis conjointement par l'INRIA Rhône-Alpes, le Laboratoire d'Informatique de Marseille et le Laboratoire d'Informatique de Grenoble. L'avantage principal d'AROM, par rapport à notre contexte de recherche vient de son système de types extensible et de son Langage de Modélisation Algébrique (LMA) qui peuvent être étendus pour offrir une gestion adaptée pour les données spatiales et temporelles. Cette extension consiste en la définition d'un ensemble de types spatiaux et temporels (AROM-ST) qui sont complétés par un ensemble d'opérateurs spatiaux et temporels.

L'objectif de notre étude est dans un premier temps de positionner AROM par rapport au standard OWL 2 (décrit dans la section 3.2.2.4). Nous mettons en évidence des points communs et des divergences qui confinent d'abord à des incompatibilités d'ordre philosophique (héritage simple versus multiple, mono versus multi-instanciation) mais vraisemblablement surmontables. Surtout, nous relevons ensuite un déficit d'AROM par rapport à OWL et à OWL 2 en termes de puissance de description. En ce qui concerne le typage, bien que OWL et OWL 2 reposent sur XML Schema [51], la plupart des outils de raisonnement associés ne prennent pas en compte les types définis par l'utilisateur. Le module de types d'AROM offre, quant à lui, une expressivité et

une souplesse supérieures dans l'intégration de nouveaux types pris en compte par AROM dès leur définition. Enfin, on peut considérer que les mécanismes d'inférences d'AROM (résolution d'équations algébriques, attachement procédural, classifications d'instance) sont complémentaires de l'ensemble des mécanismes offerts par les raisonneurs externes associés à OWL et OWL 2, qui proposent essentiellement un raisonnement à base de règles ou de tests de subsomption. Suite à cette comparaison, d'AROM et de OWL/OWL 2, nous proposons une extension du méta-modèle AROM, AROM-ONTO (motivée dans la section 5.3 et décrite dans la section 5.4), qui garantit la compatibilité du point de vue de la représentation entre AROM et OWL 2 DL et implicitement entre AROM et OWL-DL. La section 5.5 présente les mécanismes de traduction définis entre AROM-ONTO et OWL 2 DL.

5.2 Le système AROM

AROM (acronyme de *Allier Relations et Objets pour Modéliser*) [193] est un système de représentation de connaissances dont le langage est inspiré du modèle relationnel et d'UML. AROM s'inscrit dans la lignée des SRCO (Système de Représentation de Connaissances par Objets) tels que SHIRKA [208], ROME [50], FROME [75], TROPES [100] *etc.* Il en reprend les grands principes : distinction entre classes et instances, spécialisation de classes, présence de facettes de typage et d'inférence (valeur par défaut, attachement procédural, *etc.*).

AROM se distingue de ses prédécesseurs par sa manière de représenter les relations entre objets : AROM interdit l'utilisation d'attributs-liens (attributs typés par une classe), et impose, au contraire, de modéliser de manière explicite les relations entre objets à l'aide d'une seconde entité de représentation complémentaire de la structure de classe : l'*association*. Une *association* AROM est donc une entité nommée qui représente un ensemble de n-uplets d'objets appartenant aux extensions des *n classes* qu'elle relie $(n \geq 2)$. Elle est décrite par ses *rôles* (connexion entre l'*association* et l'une des *classes* reliées), à chacun desquels est associée une *multiplicité* (similaire aux cardinalités d'UML), et par un ensemble de *variables* (identiques aux variables de classes) définissant des propriétés associées aux n-uplets. Un héritage des *rôles*, *variables* et *facettes* est possible entre les *associations* à travers d'une hiérarchie de spécialisation analogue à la hiérarchie de spécialisation entre *classes*.

5.2.1 Langage de Modélisation Algébrique

Une seconde originalité d'AROM est l'utilisation d'un Langage de Modélisation Algébrique (LMA) permettant d'une part de définir les *variables* sous forme d'équations, et d'autre part d'effectuer des requêtes sur une base de connaissances. Ce langage est la transposition dans le contexte de modélisation AROM, de langages initialement introduits en *Recherche Opérationnelle* pour décrire des modèles sous forme de systèmes d'équations dans un formalisme proche des notations mathématiques. Le LMA d'AROM est, dans sa version de base, similaire au langage OCL [3] et possède une expressivité équivalente [175]. En plus d'un certain nombre d'opérateurs de base, le LMA propose des opérateurs itérés, des expressions quantifiées et des expressions de navigation pour l'accès aux objets et aux n-uplets des différentes *classes* et *associations* de la base de connaissances.

5.2.2 Le système de types

Un autre aspect important d'AROM est le fait qu'il dispose d'un module de types extensible (décrit dans le chapitre 6). Ainsi, les concepteurs ont la possibilité d'adapter le système de types d'AROM à un domaine particulier d'applications, en y ajoutant de nouveaux types et opérateurs, permettant d'intégrer plus facilement et complètement des domaines d'applications très variés. En pratique, l'extensibilité du LMA d'AROM a déjà été utilisée dans des domaines d'applications divers tels que la biologie, dans le cadre du projet Genostar[1], ou la géomatique, dans le cadre du projet GenGHIS [174]. Dans sa thèse, Bodgan Moisuc [174] a proposé AROM-ST, une extension du système de types d'AROM, qui définit un ensemble de types spatiaux et temporels. Cette extension, détaillée dans la section 6.3.1, vise également la définition d'un ensemble d'opérateurs spatiaux et temporels qui ont été intégrés dans le LMA d'AROM.

5.2.3 Description textuelle et représentation graphique de bases de connaissances

Le modèle de représentation des connaissances AROM s'appuie sur un langage de description textuelle, dont la BNF (Backus Naur Form) est définie dans [45]. Cette formalisation BNF est reprise et complétée, dans le cadre de cette thèse, afin d'intégrer les nouvelles structures de représentation proposées dans la section 5.4. Elle peut être consultée dans l'annexe A. Toute base de connaissances AROM peut donc être décrite en utilisant le langage textuel, et peut être sauvegardée dans ce format, appelé format `txta` (texte AROM). La figure 5.1 illustre une base de connaissances nommée `Enseignement`, exprimée dans le format `txta`, et qui modélise de manière très simplifiée le système des cours dans l'Enseignement Supérieur en France.

Cette base de connaissances contient une classe `Enseignant` décrite par les variables `numInsee` (numéro INSEE), `service` (nombre d'heures enseignées), `salaire` (salaire mensuel), et `salaireHoraire` (salaire horaire). Les enseignants se divisent en temporaires (classe `Temporaire`) et permanents (classe `Permanent`) pour lesquels un service minimum (`serviceMin`) et un salaire fixe (`salaireFixe`) sont définis. L'extrait montre également trois associations. L'association `Enseigne` lie la classe Enseignant à deux autres classes : `Formation` et `Cours`. Les contraintes de multiplicité expriment qu'un enseignant peut donner un nombre quelconque de cours dans un nombre quelconque de formations. L'association `Dirige` modélise le fait qu'un permanent peut diriger au plus une formation. `Dirige` se spécialise en `Dirige3eCycle` pour signifier que seul un `Professeur` (une autre classe de la base) peut diriger une formation de troisième cycle, fonction pour laquelle il reçoit une prime.

AROM dispose également d'un environnement visuel de conception, l'IDE (*Integrated Development Environment*), basé sur une représentation graphique similaire aux diagrammes de classes d'UML. En effet, les notations graphiques utilisées dans AROM (voir figure 5.2) sont un sous-ensemble des notations UML pour les diagrammes de classes, avec quelques modifications qui dénotent les différences entre les modèles classes/associations de UML et d'AROM. Les classes sont représentées par des rectangles. Les associations sont représentées par des rectangles grisés aux coins arrondis, avec le nom des rôles et leur multiplicité placés sur l'arête connectant l'association et la classe correspondante. Ces notations graphiques sont supportées par l'environnement de développement d'AROM, appelé IME (Interactive Modelling Environment). Il permet à un utilisateur/concepteur d'une base de connaissances d'esquisser rapidement l'ébauche du schéma de

[1] http ://www.genostar.com/

cette base, et d'y apporter les modifications fréquentes en phase de conception [101]. Une fois construite sous l'IME, une base peut être exportée au format `txta`. La représentation graphique de la base `Enseignement` est visible sur la figure 5.2.

```
knowledge-base: Enseignement

class: Enseignant
  variables:
    variable: numInsee
      type: string
      documentation: "numéro INSEE"
    variable: service
      type: float
      documentation: nombre d'heures enseignées"
    variable: salaire
      type:float
      domain: [0.0 .. 10000.0]
      unit: "€"
    variable: salaireHoraire
      type: float

class: Temporaire
  super-class: Enseignant
  variables:
    variable: salaire

class: Permanent
  super-class: Enseignant
  variables:
    variable: serviceMin
      type: integer
      default: 192
      documentation: "nombre d'heures min. à
effectuer"
    variable: salaireFixe
      type:float
…

association: Enseigne
  roles:
    role: enseignant
      type: Enseignant
      multiplicity: min:0 max:1
    role: cours
      type: Cours
      multiplicity: min:0 max:*
    role: formation
      type: Formation
      multiplicity: min:0 max: *
  variables:
    variable: volumeHoraire
      type: integer

association: Dirige
  roles:
    role: directeur
      type: Permanent
      multiplicity: min:1 max:1
    role: formation
      type: Formation
      multiplicity: min:0 max:1

association: Dirige3eCycle
  super-association: Dirige
  roles:
    role: directeur
      type: Prof
    role: formation
      type: Form3eCycle
  variables:
    variable: prime
      type:float
```

FIG. *5.1: Exemple de la description textuelle dans la syntaxe AROM d'une base de connaissances, appelée Enseignement.*

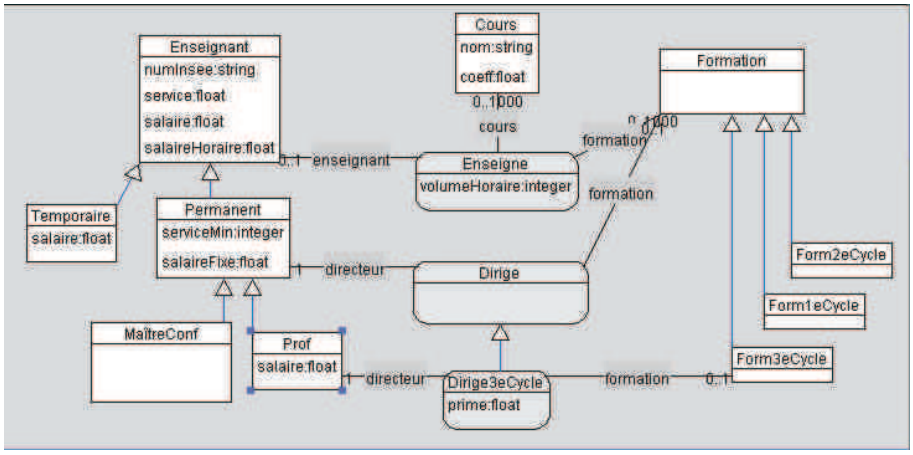


FIG. *5.2: Représentation graphique de la base de connaissances Enseignement.*

5.2.4 Plate-forme AROM

La représentation des connaissances en AROM est mise en œuvre à travers un environnement réalisé en langage Java, et appelé *système AROM*. Le système AROM assure la représentation,

sous la forme d'objets informatiques, des entités du *modèle* AROM (formalisme de représentation des connaissances proposé par AROM et reposant sur les structures de classes et d'associations). La plate-forme AROM, dont l'architecture est illustrée dans la figure 5.3, est la partie applicative d'AROM qui englobe le système AROM [101]. Située au dessus du noyau que forme le système, la plate-forme AROM propose un certain nombre de bibliothèques pour l'exploitation des bases de connaissances. Chacune de ces bibliothèques a été réalisée à l'aide de l'API Java AROM et propose sa propre API. Parmi ces bibliothèques, `AROMClassif` est une API pour la classification d'objets et de tuples avec propagation, `AROMQuery` est une API pour formuler des requêtes sur une base AROM, `XAROM` est une API permettant d'exporter des bases AROM au format XML (voir figure 5.3).

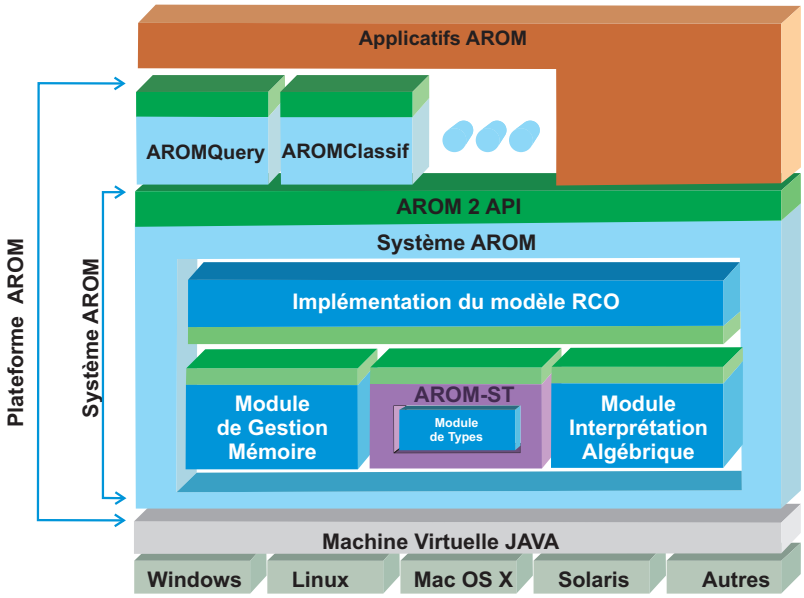


FIG. *5.3: La plate-forme AROM.*

Le système AROM est lui-même conçu de façon modulaire, chaque module étant en charge d'une fonction précise du système. Le module de gestion mémoire (voir figure 5.3) assure l'accès en mémoire aux instances d'objets AROM ; le module de types définit l'ensemble des types reconnus dans une base AROM et les opérations possibles sur ces types [48] ; le module d'interprétation algébrique assure l'interprétation d'expressions du Langage de Modélisation Algébrique (LMA) d'AROM. Chaque module ne communique avec les autres modules qu'au travers d'une API interne clairement isolée.

5.2.5 La classification en AROM

La classification est un mécanisme d'inférence central dans le système AROM. En toute généralité, la classification (d'une instance ou d'une classe) permet d'expliciter des connaissances (jusque là implicites) sur l'instance ou la classe [101]. Dans le cas de la classification d'instance, on cherche à en obtenir une caractérisation plus précise, en déterminant les classes les plus spécialisées auxquelles l'instance peut être rattachée. Pour cela, il s'agit d'établir le degré de ressemblance entre l'instance et une classe candidate [101]. Les classes candidates sont a priori toutes les sous-classes de la classe actuelle d'attachement de l'instance. Dans le cadre d'AROM, il faut tenir

compte de la présence de deux entités duales : les classes et les associations. On est donc amené à classer les instances de celles-ci : les objets et les tuples.

La classification d'une instance (d'un objet ou d'un tuple) repose sur deux types de vérification : l'une appelée *attachement statique*, l'autre *attachement dynamique* [101].

Attachement statique

La vérification statique de l'attachement d'un objet ou d'un tuple s'assure que la valeur de chaque variable de l'objet ou du tuple est bien du type attendu. A l'issue de cette étape, la classe ou l'association candidate au rattachement est temporairement marquée :

- *Admitted*, si chaque variable de la classe ou de l'association a une valeur bien typée dans l'objet ou le tuple ;
- *Eligible*, si elle peut être marquée comme *Admitted* au regard d'un sous-ensemble des variables de la classe ou de l'association, les valeurs de toutes les autres variables dans l'objet ou le tuple étant inconnues ;
- *Rejected*, si au moins une variable de la classe ou de l'association a une valeur, dans l'objet ou le tuple, qui n'est pas du type attendu.

Pour les tuples, une vérification spécifique est lancée pour les rôles, qui s'assure que chaque valeur objet d'un rôle est bien de la classe attendue (ou d'une sous-classe de celle-ci). Si l'objet est d'une super-classe de la classe attendue alors une classification de cet objet doit être lancée pour déterminer si l'objet peut descendre vers la classe attendue. Si la valeur-objet n'est pas de la classe (ou sous-classe) attendue, l'association est marquée *Rejected*. Enfin, les contraintes de multiplicité sont vérifiées pour valider définitivement l'étiquetage de l'association.

Attachement dynamique

Une fois la vérification statique d'attachement effectuée, une vérification dynamique est lancée seulement pour toute classe ou association temporairement marquée *Admitted* ou *Eligible*. Elle consiste à :

a) tester si les valeurs des variables et des rôles satisfont dynamiquement les contraintes ;

b) tester, dans le cas d'un objet, la validité des liens dans lesquels il est impliqué ;

c) lancer les mécanismes d'inférence disponibles (attachement procédural, définition) et vérifier que le résultat obtenu est bien du type attendu.

Si lors de ces trois étapes, une incohérence est détectée, la classe ou l'association est définitivement marquée *Rejected*, sinon le marquage temporaire - *Admitted* ou *Eligible* - est définitivement acquis.

5.3 De AROM vers AROM-ONTO : motivations

AROM est un outil générique de modélisation et d'exploitation des connaissances, qui peut être utilisé pour la représentation des ontologies au sens large du terme. Son intérêt vient de ses capacités de modélisation, de son expressivité et sa simplicité, ainsi que de l'ensemble d'inférences que l'on peut considérer comme complémentaires de celles offertes par les *Logiques de*

Description, notamment la classification des instances mais aussi la valeur par défaut, l'héritage et l'attachement procédural.

AROM offre donc un langage de description de connaissances qu'il paraît intéressant de comparer à OWL et à OWL 2, en leur qualité de standards pour la représentation d'ontologies. OWL et OWL 2 sont la synthèse de travaux de recherche fondamentales menées depuis plus de 30 ans autour des Logiques de Description, et sont destinés à faciliter l'annotation de ressources dans le cadre ouvert du Web Sémantique. Ils sont très expressifs et disposent d'un éventail d'outils de modélisation et de raisonnement (voir sections 3.2.3 et 3.2.4). Cependant, pour l'exploitation des ontologies, on dispose pour l'instant seulement de la classification des concepts comme principal moyen de raisonnement. On constate aussi que d'autres lacunes, parmi lesquelles figurent la gestion incomplète des domaines concrets (types de données), l'absence de relations n-aires, l'impossibilité d'importer sainement des programmes externes, l'impossibilité de traiter la coopération entre informations symboliques et numériques, *etc.*, rendent difficiles la création et l'utilisation d'ontologies pour des applications concrètes. C'est le cas, par exemple, de l'interprétation d'images satellitaires - pour des diagnostics environnementaux - qui nécessite la prise en compte d'informations numériques dans un environnement symbolique et la représentation de structures spatiales complexes [148].

Parmi les avantages à tirer de l'utilisation du système AROM, on peut citer notamment :

- son approche objet/relations n-aires, similaire aux diagrammes de classes UML ;
- son système de types extensible, contrôlant la représentation, au sein d'une base de connaissances, de connaissances calculées extérieurement ;
- son langage de modélisation algébrique intégré qui permet l'écriture déclarative d'équations et de contraintes algébriques entre et au sein des objets, ainsi que la formulation de requêtes sur le contenu d'une base de connaissances ;
- son classifieur d'objets et de relations ;
- la lisibilité de ses modèles, ce qui provient essentiellement du paradigme objet ;
- son implémentation robuste et son API Java qui facilite l'exploitation des connaissances depuis le vaste monde de la programmation.

Cependant, il est patent que plusieurs inconvénients majeurs limitent l'utilisation pérenne de AROM, parmi lesquels : *i)* le manque de capacités de représentation pourtant essentielles : l'héritage multiple et la multi-instanciation en premier lieu, mais aussi *ii)* le manque d'une algèbre de relations, même simpliste, qui permettrait une représentation riche des relations entre objets. Il faut néanmoins, rappeler qu'une étude ainsi qu'une définition complètes de la relation de composition en AROM ont été proposées [102].

Ainsi, les forces de AROM paraissent être les faiblesses de OWL (et de l'évolution OWL 2) et *vice-versa*. De ce fait, comment rendre ces deux langages, et les systèmes qui les embarquent, complémentaires ? Comment les faire coopérer pour tirer partie des atouts de l'un et de l'autre ? Nous proposons de répondre à ces questions en deux étapes :

a) en proposant une extension du méta-modèle d'AROM qui intègre les constructions validées par OWL 2. La nouvelle extension, nommée AROM-ONTO et présentée dans la section 5.4, garantit la compatibilité du point de vue de la représentation entre les bases de connaissances AROM et les ontologies OWL 2.

b) en définissant un traducteur de AROM-ONTO vers OWL/OWL 2 et réciproquement (voir figure 5.4). Ainsi, une ontologie OWL 2 DL (respectivement, une base de connaissances

AROM-ONTO) pourra être transcrite en une base de connaissances AROM-ONTO (respectivement, en une ontologie OWL 2 DL) et l'on pourra bénéficier des inférences mises à disposition par le système AROM-ONTO (respectivement, des outils et des raisonneurs construits pour OWL). Ce traducteur est présenté dans la section 5.5.

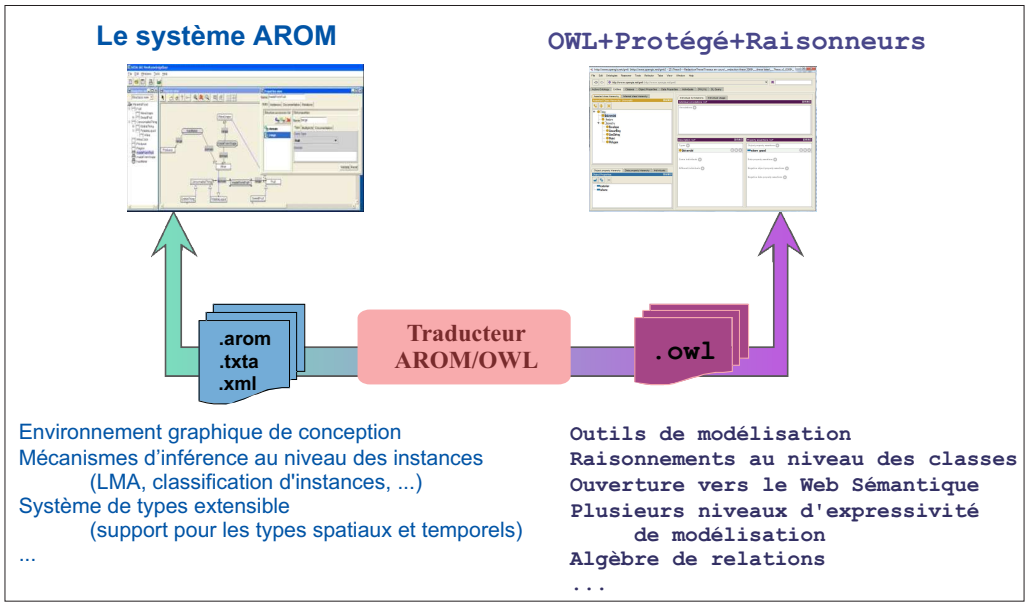


FIG. *5.4: Illustration de notre approche.*

5.4 L'extension AROM-ONTO

L'extension AROM-ONTO est issue d'un travail de rapprochement entre AROM et OWL DL, initié en 2005 par nos soins et décrit en détail dans [168, 167]. Ici, nous prolongeons ce travail, en faisant une comparaison entre les caractéristiques de AROM et celles de OWL 2 DL, selon trois points de vue différents : la représentation, le typage et les inférences qui peuvent être réalisées à partir des connaissances modélisées.

Cette extension est un prérequis, à la fois à l'importation dans AROM d'une ontologie OWL 2 DL, et à l'exportation vers une ontologie OWL 2 d'une base de connaissances AROM, avec un minimum de perte d'informations. Du point de vue de la représentation, la comparaison a permis d'identifier les points communs et les différences entre les deux sous-langages (OWL 2 DL et OWL 2 Full) du standard OWL 2 et le langage AROM. Dans la section suivante, nous présentons les différences majeures qui existent entre AROM et OWL 2 DL, ainsi que l'extension qui a été proposée au niveau du méta-modèle d'AROM pour garantir une compatibilité maximale en termes de représentation avec le langage OWL 2. Nous ne cherchons pas à résoudre toutes les incompatibilités, mais simplement à dégager les pistes les plus importantes pour faire évoluer AROM le plus favorablement possible vers le Web Sémantique, en étudiant les avantages et les inconvénients d'une expressivité élevée.

5.4.1 Similarités de représentation entre AROM et OWL 2

La comparaison des deux langages montre, dans un premier temps, certaines similarités entre leurs primitives de représentation. Ainsi, une ontologie OWL 2, comme une *base de connaissances* AROM, représente un cadre pour l'introduction des définitions, des axiomes et des assertions visant le domaine modélisé.

Ontologies et bases de connaissances

Une ontologie OWL 2 est construite à partir d'un ensemble d'*entités* : les *classes*, les *propriétés-lien*, les *attributs*, les *types de données*, les *propriétés d'annotation* et les *individus nommés* (voir figure 5.5).

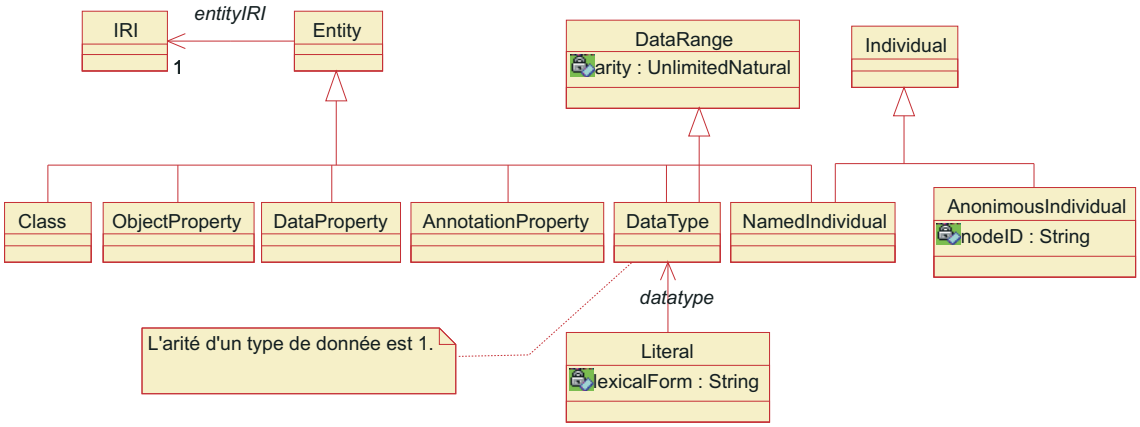


FIG. *5.5: Les briques fondamentales du langage OWL 2 : les entités, les littéraux et les individus anonymes (source [181]).*

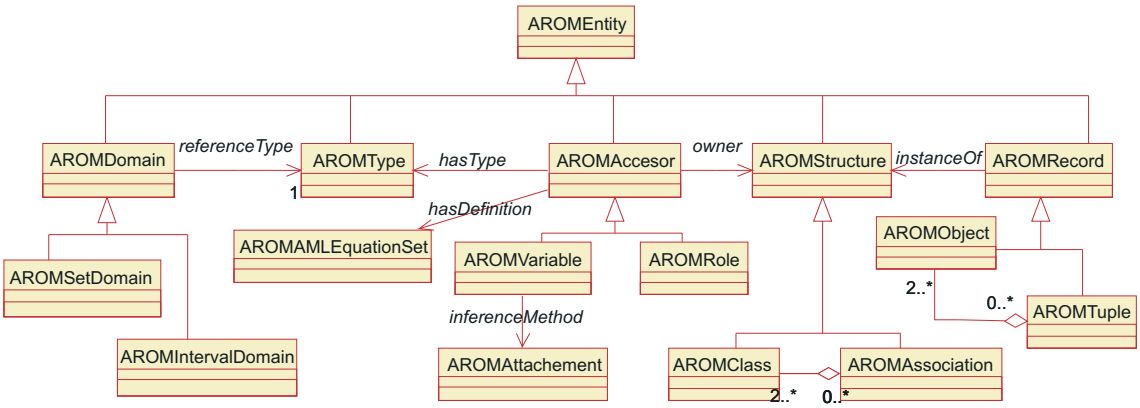


FIG. *5.6: Les briques fondamentales du langage AROM : les entités, les équations du LMA et les primitives pour l'attachement procédural.*

Une *base de connaissances* AROM est comparable à une ontologie, dans le sens où, elle représente un modèle conceptuel mettant en œuvre les concepts (classes) et les relations (associations) du domaine d'application étudié, ainsi que l'ensemble des individus qui font partie de ce domaine (voir figure 5.6). Par la suite, nous présentons les correspondances qui existent entre les entités AROM et les entités OWL 2.

Classes

Les *concepts* AROM sont modélisés par des *classes* qui possèdent une *intension* et une *extension* [193]. L'intension[2] d'une classe est spécifiée à l'aide de *variables*. OWL 2 suit le même principe avec la seule différence que les *attributs* des concepts ne sont pas inclus dans la structure de la *classe* mais sont conçus comme des entités à part : les *propriétés-lien*. La figure 5.7 illustre le concept de département défini en OWL 2 et en AROM par des classes `Department`. On peut facilement observer la correspondance entre la *variable* `surface` en AROM et l'*attribut* homonyme en OWL 2, pour modéliser la même *caractéristique* de la classe `Department`.

Une variable AROM est décrite par trois facettes : une *facette de documentation* (`documentation`), une *facette de type* (`domain`) et une *facette d'inférence* (`definition`). Ainsi, en AROM, il est possible de restreindre les valeurs d'une *variable* à travers la *facette de type* qui permet de spécifier la palette de valeurs d'un type donnée qui peuvent être attribuées à la variable en question. La déclaration équivalente, en OWL 2, est la restriction de type de données. Un exemple de restriction de type définie dans les deux formalismes est présenté sur la figure 5.7. Il s'agit de l'attribut `population`, dont les valeurs sont comprises dans l'intervalle d'entiers [1, 2000000]. Les valeurs possibles de la variable homonyme en AROM sont spécifiées à travers la *facette de type* `domain`.

En AROM, il est également possible de définir la valeur d'une variable à travers une équation algébrique construite à partir d'autres variables définies dans la même base de connaissances, et d'un ensemble d'opérateurs (pour plus de détails, voir [192]). Un exemple de telle définition est illustré sur la figure 5.7. Il s'agit de la variable `cropArea`, définie comme la différence entre la surface totale du département et la surface forestière du même departement. En revanche, la définition de la variable `cropArea` reste basique en OWL 2 et ne traduit pas la spécification mathématique.

OWL 2

```
Declaration(Class(Department))

Declaration(DataProperty(population))
DataPropertyDomain(population Department)
DataPropertyRange(population
    DatatypeRestriction(xsd:integer
                        maxInclusive
                        "2000000"^^xsd:integer))
DataPropertyRange(population
    DatatypeRestriction(xsd:integer
                        minInclusive
                        "1"^^xsd:integer))

Declaration(DataProperty(surface))
DataPropertyDomain(surface Department)
DataPropertyRange(surface xsd:float)

Declaration(DataProperty(forestArea))
DataPropertyDomain(forestArea Department)
DataPropertyRange(forestArea xsd:float)

Declaration(DataProperty(cropArea))
DataPropertyDomain(cropArea Department)
DataPropertyRange(cropArea xsd:float)
```

AROM

```
class : Department
variables :
variable : population
    type : integer
    domain: [1..2000000]
variable : surface
    type : float
variable : forest Area
    type : float
variable: crop Area
    type: float
    definition: cropArea=surface-forestArea
```

FIG. *5.7: Le concept de département défini en OWL 2 et en AROM à travers la classe Department.*

[2]La description en *intension* d'une classe consiste en un ensemble de "propriétés", qui s'expriment par l'appartenance d'un *attribut* à un *domaine* [83].

OWL 2 fournit, pour chaque ontologie, deux classes prédéfinies : `owl:Thing` une classe générique spécialisée par toutes les autres classes et qui n'accepte pas de généralisation, et `owl:Nothing` une classe vide, qui spécialise toutes les autres classes et qui n'accepte pas de spécialisation. Pour des raisons de compatibilité, nous avons introduit dans le modèle de AROM-ONTO deux classes prédéfinie `Thing` et `Nothing` qui correspondent de près aux classes `owl:Thing` et `owl:Nothing`, et dont la sémantique est définie dans le tableau B.4 de l'annexe B.

Propriétés-lien et associations

En ce qui concerne les relations, les différentes liaisons entre les classes OWL 2 sont définies à l'aide de *propriétés-lien*, équivalentes aux *associations* binaires d'AROM. La figure 5.8 donne la définition de la relation `livesAt` dans les deux formalismes. Les classes reliées sont désignées par les primitives `rdfs:domain` et `rdfs:range` en OWL 2, et par les *rôles* homonymes en AROM. En OWL 2, on a la possibilité de spécifier la cardinalité d'une propriété à travers des restrictions de cardinalité imposées aux classes désignées par le domaine respectivement l'image de la propriété respective (voir figure 5.8). Une définition équivalente en AROM est réalisée à travers les *facettes de multiplicité* attachées aux rôles de l'association que l'on souhaite contraindre (voir figure 5.8).

OWL 2

```
Declaration(Class(Address))
SubClassOf(Address ObjectMinCardinality(1 livesAt Address))

Declaration(Class(Person))
SubClassOf(Person ObjectMinCardinality(1 livesAt Person))

Declaration(ObjectProperty(livesAt))
ObjectPropertyDomain(livesAt Person)
ObjectPropertyRange(livesAt Address)
```

AROM

```
association: livesAt
super-association: where
roles:
    role: domain
       type: Person
       multiplicity: min:1
    role: range
       type: Address
       multiplicity: min:1
variables:
    variable: functional
       type: boolean
       definition: functional=true
```

FIG. *5.8: La relation livesAt, définie en AROM et en OWL 2.*

OWL 2 n'offre pas les moyens de définir une relation *n*-aire entre classes. Noy *et al.* proposent de représenter une relation *n*-aire par une classe à laquelle sont attachées *n* propriétés-lien qui désignent les rôles [189]. Cette confusion de structures de représentation s'avère difficile à gérer par les raisonneurs et à comprendre par les utilisateurs.

Les associations AROM peuvent se décrire à l'aide de *variables*. Par exemple, pour une association `livesAt`, définie entre les classes `Person` et `Address`, on peut déclarer les *variables* `startDate` et `endDate` qui indiquent la période durant laquelle une personne habite à une adresse donnée. À ce sujet, le langage OWL 2 offre la possibilité d'utiliser le même IRI, *I*, pour identifier différentes type d'entités (*i.e.* une propriété et un individu), comme illustré dans la figure 5.9. Cet usage de *I* est souvent appelé méta-modélisation, car il peut servir à réaliser des déclarations sur les classes et les propriétés elles-mêmes [181]. Dans ce cas, les entités identifiées par *I* (*i.e.* la propriété `livesAt` et l'individu `livesAt`) sont considérées comme des *vues* (*views*) différentes de la même notion, identifiée par *I* (*i.e.* `livesAt`). Néanmoins, si l'ontologie contenant une telle déclaration est interprétée selon la *sémantique directe* (*i.e.* l'ontologie est OWL 2 DL), le lien entre l'individu `livesAt` et la propriété `livesAt` est perdu, car les deux *views* sont interprétées

de façon indépendante, par rapport à deux univers d'interprétation différents.

OWL 2

```
Declaration(ObjectProperty(livesAt))
ObjectPropertyDomain(livesAt Person)
ObjectPropertyRange(livesAt Address)

Declaration(DataProperty(startDate))
DataPropertyRange(startDate xsd:dateTime)

Declaration(Individual(livesAt))
DataPropertyAssertion(startDate livesAt "2009-07-19T21:30:00"^^xsd:dateTime)
```

FIG. *5.9: Exemple de méta-modélisation en OWL 2.*

Certaines caractéristiques prédéfinies peuvent être spécifiées pour les propriétés-lien de OWL 2, comme la *transitivité*, la *symétrie*, l'*asymétrie*, l'*unicité*, l'*unicité inverse*, la *réflexivité* et l'*irréflexivité*. Si elles ne trouvent pas de correspondant direct en AROM, elles peuvent toutefois être simulées à l'aide des variables d'association. Si on prend l'exemple de la figure 5.8, l'association `livesAt` est définie comme étant *unique* par l'intermédiaire de la variable `functional`. La définition d'*unicité* est prise en compte par les raisonneurs d'OWL 2 qui peuvent en déduire les conséquences logiques et les implications vis-à-vis de la cohérence de l'ontologie. La solution que nous proposons pour AROM-ONTO est de séparer la définition des associations en associations binaires (`2-Association`) et association n-aires (`n-Association`) (voir figure 5.10). Les associations binaires sont spécialisées par les propriétés (`AROMProperty`). Les caractéristiques des propriétés-lien définies par OWL 2 sont modélisées comme des méta-variables `booléennes` prédéfinies. Pour chaque nouvelle méta-variable, nous avons spécifié les implications logiques qu'elle impose pour les associations dans lesquelles elle est activée (a la valeur vrai). Contrairement aux associations binaires, les associations de type `AROMProperty` ne peuvent pas être décrites par des variables (d'où la cardinalité 0 pour la relation entre les méta-classes `AROMProperty` et `AROMVariable`) définies par l'utilisateur et leurs deux rôles sont prédéfinis (`property-domain` et `property-range`).

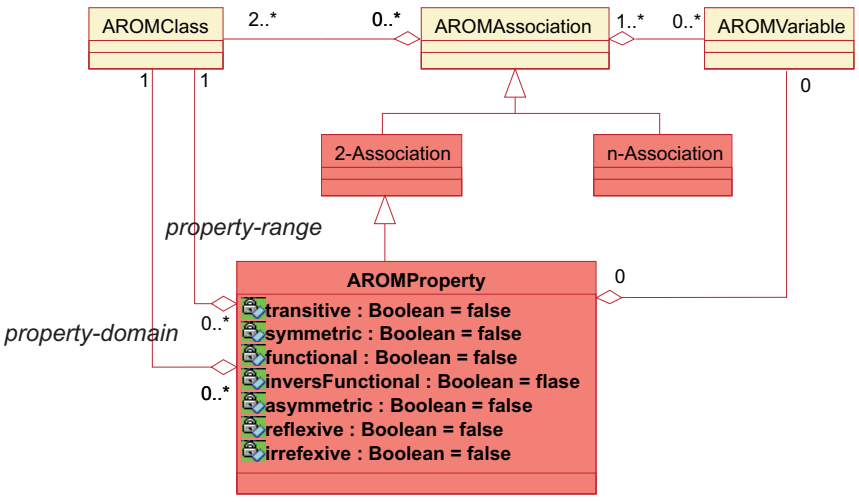


FIG. *5.10: Introduction de nouvelles structures au niveau du mèta-modèle d'AROM-ONTO,* `n-Association`*,* `2-Association`*,* `AROMProperty`*.*

Nous avons étendu la syntaxe du langage AROM-ONTO par une nouvelle primitive, `property`, qui facilite la définition des propriétés. Un exemple de définition d'une propriété en AROM-ONTO est illustré par la figure 5.11. La spécification BNF complète pour cette nouvelle primitive est présentée dans l'annexe A. Nous avons également étendu en conséquence la fonction d'interprétation d'une base de connaissances afin de garantir une interprétation adaptée des connaissances définies à travers la nouvelle structure (voir annexe B).

OWL 2 définit deux propriétés valables pour toutes les ontologies : `owl:topObjectProperty` et `owl:bottomObjectProperty` qui modélisent respectivement une propriété générique spécialisée par toutes les autres propriétés et n'acceptant pas de généralisation et une propriété vide qui spécialise toutes les propriétés d'une ontologie, et qui n'accepte pas de spécialisation. À l'instar de OWL 2, nous avons introduit dans le modèle de AROM-ONTO deux propriétés prédéfinies `TopProperty` et `BottomProperty` qui correspondent de près aux propriétés prédéfinies de OWL 2. Leur sémantique est définie dans le tableau B.5 de l'annexe B.

Les attributs de OWL 2 disposent d'une seule caractéristique : l'*unicité*. Les correspondants des attributs en AROM sont les variables, qui peuvent être *simples* ou *multivaluées*. Pour les variables simples, une instance ne peut pas avoir plusieurs valeurs, donc l'*unicité* est implicite pour les variables de ce type. Les variables multivaluées sont, par principe, non-unique.

```
AROM-ONTO
property: hasMainResidence
 super-property: hasResidence
 property-domain
    type: Person
property-range
    type: Address
 functional=true
```

FIG. *5.11: Exemple de définition de propriété en AROM-ONTO.*

Propriétés d'annotation et facettes de documentation

En plus de *propriétés-lien* et d'*attributs*, OWL 2 dispose de *propriétés d'annotation* pour attacher des informations additionnelles aux `IRI`, aux individus anonymes et aux littéraux et de *propriétés d'ontologie* pour définir les compatibilités entre plusieurs versions d'une ontologie. Toutes ces propriétés peuvent être modélisées en AROM-ONTO par des variables prédéfinies de type `string` ou `IRI`, attachées aux structures, comme illustré dans la figure 5.12. OWL 2 offre également des primitives pour la définition des axiomes de subsomption entre les *propriétés d'annotation*. Cependant, les annotations ne sont pas prises en compte par les interprétations et sont, en conséquence, ignorées dans les raisonnements.

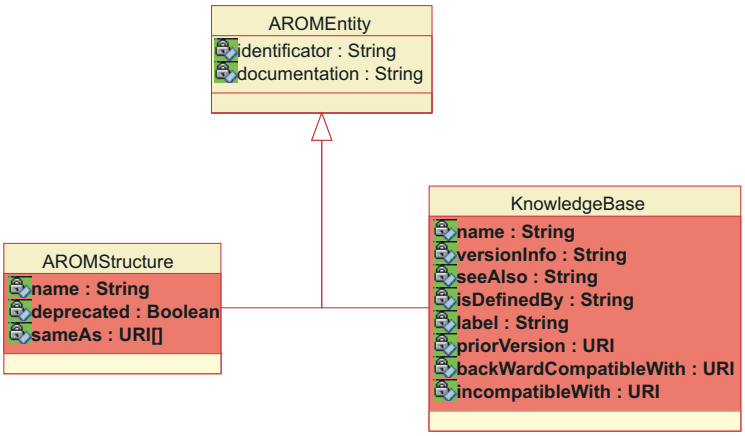


FIG. *5.12: Les variables prédéfinies ajoutées pour les méta-classes AROMStructure et KnowledgeBase.*

En AROM-ONTO, une facette de documentation peut être attachée à toute entité. Celle-ci peut-être divisée en deux parties : l'une au format texte et l'autre au format IRI. Seul l'un des deux formats, ou les deux simultanément, peuvent être spécifiés pour caractériser une entité. Pour les classes, les associations et les propriétés, la facette de documentation peut être différente d'une niveau de spécialisation à un autre. Ceci permet d'attribuer une documentation de plus en plus précise tout au long de la hiérarchie de spécialisation (de classe, d'association ou de propriété). La règle de spécialisation de la facette de documentation impose qu'une facette de documentation masque la ou les documentations qu'elle spécialise [45]. Néanmoins, AROM-ONTO ne permet pas de définir des annotations sur des annotations ni de spécialiser les annotations attachées aux individus (objets et tuples) et aux variables.

Individus

À travers des déclarations OWL 2, on peut introduire des individus dans une ontologie. Ceux-ci sont équivalents aux objets qui existent en AROM. Les individus et les objets sont des entités du monde réel, des instances modélisées par les concepts. Ainsi, les classes correspondent aux ensembles de choses qui apparaissent naturellement dans un domaine d'étude et les individus désignent les entités concrètes qu'on peut regrouper dans ces classes.

Les individus de OWL 2 peuvent être nommés, ils sont alors identifiés par un IRI, ou anonymes, et dans ce cas, ils ne disposent pas d'un identifiant global et peuvent être utilisés exclusivement dans l'ontologie qui les introduit. AROM impose l'existence et l'unicité des identificateurs des individus. Le système considère distinctes deux instances ayant des identificateurs différents sans faire de tests supplémentaires sur leurs contenus respectifs. Sur le Web, un tel postulat est impossible. Par exemple, on pourrait désigner la même personne de plusieurs façons différentes (*i.e.* en utilisant des IRI différents). En conséquence, le langage OWL 2 ne fait aucune supposition sur les individus en l'absence de déclarations explicites les concernant. Pour faire des assertions sur l'identité d'individus, OWL 2 dispose de deux axiomes dédiés : `SameIndividual` et `DifferentIndividuals`.

L'axiome `SameIndividual` s'emploie pour affirmer que deux ou plusieurs adresses IRI exposent le même individu. L'axiome `DifferentIndividuals` est utilisé pour l'affirmation contraire : deux ou plusieurs IRI se rapportent à des individus différents. Comme mentionné précédemment, AROM suppose tous les individus différents. En effet, à l'intérieur d'une base de connaissances,

il n'y a pas de raison de faire des assertions redondantes qui déclarent l'existence d'un même individu plusieurs fois. Néanmoins, dans la perspective de l'introduction d'un mécanisme d'importation d'ontologies ou d'autres bases de connaissances, il est nécessaire de pouvoir définir les individus qui sont égaux et ceux qui sont différents. Ces affirmations constituent des bases importantes pour les raisonnements.

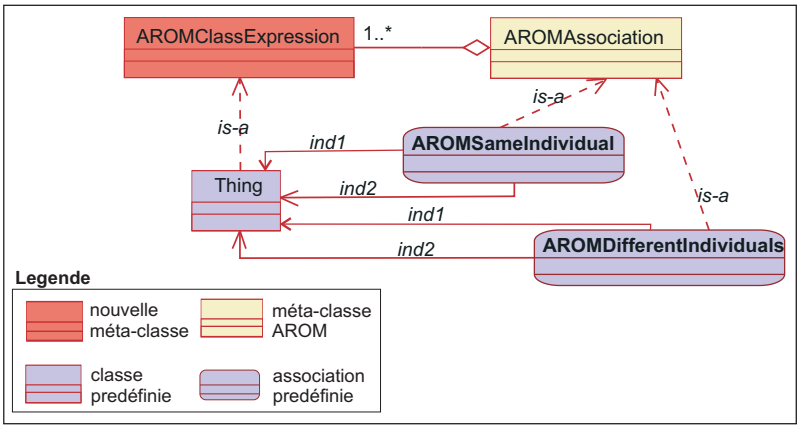


FIG. *5.13: La définition en AROM-ONTO de l'équivalence et la disjonction entre objets.*

Pour répondre à ces besoins, nous avons ajouté au modèle de AROM-ONTO deux associations binaires prédéfinies, `AROMSameIndividual` et `AROMDifferentIndividuals`, qui peuvent être instanciées sans redéfinition dans toute base de connaissances. Ainsi, chaque axiome `SameIndividual` défini dans OWL 2 pour un ensemble de k individus, est équivalent à $\frac{k(k-1)}{2}$ tuples de l'association `AROMSameIndividual`. Cela peut s'avérer difficile à gérer à la main, mais ce choix est motivé par le fait qu'AROM ne permet pas la définition des associations ayant une cardinalité variable. En d'autre termes, le nombre d'objets reliés par les tuples d'une même association ne peut pas varier. La classe attachée à chaque rôle de ces associations, est la même : `Thing`, la classe la plus générale du modèle.

Ces deux associations ont une sémantique fixée, décrite dans le tableau B.7 de l'annexe B. Néanmoins, les tuples des associations `AROMSameIndividual` et `AROMDifferentIndividual` sont, à ce jour, utilisés principalement pour garantir une certaine compatibilité avec les déclarations de OWL 2 DL. En d'autres termes, ils permettent de garder la trace des déclarations correspondantes en OWL 2. Mais, le système AROM-ONTO ne cherche pas à déterminer les conséquences logiques d'une déclaration d'égalité/de disjonction entre individus. Simplement, les algorithmes d'analyse sémantique présentés dans le chapitre 7 exploitent ces tuples sans remettre en cause la connaissance qu'ils portent, pour créer des ponts entre les individus définis dans des ontologies différentes et calculés/déclarés comme égaux suite à un processus d'alignement d'ontologies (processus externe à ONTOAST).

Comme perspective de ce travail, il serait intéressant de réaliser une étude approfondie sur les problèmes de cohérence qui peuvent apparaître dans une base de connaissances contenant des tuples des associations `AROMSameIndividual` et `AROMDifferentIndividual`. Il s'agit notamment de situations où le même tuple $\langle x,y \rangle$ appartient à l'extension des deux associations. Il se peut également qu'un tuple $\langle x,y \rangle$ soit attaché à l'association `AROMSameIndividual`, alors que les objets x et y appartient à des classes définies comme complémentaires ou disjointes. Le comportement du système AROM-ONTO face à ce type d'incohérence reste donc à définir.

5.4.2 Incompatibilités majeures de représentation entre AROM et OWL 2

Importation

Le langage OWL 2 a été conçu pour un environnement distribué par excellence et, de ce fait, a intégré la possibilité d'étendre des ontologies existantes, ou d'employer les concepts décrits dans des ontologies externes pour compléter la définition d'une nouvelle ontologie [143]. À cette fin, on utilise des IRIs pour définir des espaces de nommage indiquant à quelle ontologie se rapporter lorsqu'on parle de concepts externes à l'ontologie courante. L'importation est un mécanisme transitif [181].

Le système AROM ne permet pas de référencer des concepts ou des associations définis au-delà de la base de connaissances courante. De plus, le type `IRI` n'existe pas parmi l'ensemble de types de base offerts par AROM. Cependant, définir un nouveau type en AROM est possible. Il suffit en effet, d'utiliser une bibliothèque Java qui déclare le type IRI (comme celle définie dans le paquetage `com.hp.hpl.jena.IRI`[3]) et de l'étendre pour qu'elle fournisse les restrictions de typage de variables imposées par AROM. À l'image de Protégé [7] et autres éditeurs d'ontologies ou raisonneurs, AROM-ONTO est capable de chercher les bases de connaissances définies ailleurs et d'établir des liaisons entre les concepts qu'elles contiennent et ceux de la base de connaissances courante [168]. À ces fins nous avons étendu la syntaxe du langage AROM-ONTO, comme illustré dans l'annexe A, avec les primitives nécessaires pour la définition des importations.

Mono - instanciation vs. multi - instanciation

On pense habituellement aux instances des classes OWL 2 comme à des individus dans notre univers de choses. Il suffit de déclarer un individu comme membre d'une classe pour l'introduire. Il n'est même pas obligatoire de définir une classe pour introduire une instance, car OWL dispose de la classe universelle `Thing`. Donc, on peut facilement déclarer l'existence de n'importe quel objet en le liant à la classe la plus générique possible. De plus, en OWL 2, un même individu peut appartenir à l'extension de plusieurs classes qui n'ont pas été explicitement définies comme disjointes.

Ce n'est pas le cas en AROM qui considère une classe comme un moule pour la définition des instances. Celles-ci ne possèdent pas forcément une description complète au moment de leur création mais sont attachées exclusivement à la classe qui les introduit. Néanmoins, lorsque le contenu d'un objet change ou se précise, il peut migrer vers d'autres classes, notamment vers des spécialisations de sa classe d'introduction. La mono-instanciation limite considérablement le pouvoir d'expression d'AROM et pose des problèmes importants lors de l'introduction des classes complexes (*i.e.* définies à l'aide des opérateurs ensemblistes). L'opération d'intersection de deux ou plusieurs classes n'a pas grand sens en AROM du point de vue des extensions de classes qui sont toujours disjointes : une instance (objet ou tuple) appartenant à une seule structure (classe ou association) à un moment donné, toute intersection de classes sera vide [168, 167]. Les mêmes raisons nous empêchent de parler de classes énumérées, complémentaires, *etc.*

[3] http ://stderr.org/doc/libjena-iri-java/javadoc/com/hp/hpl/jena/iri/IRI.html

La spécialisation et l'héritage

Le principe de la spécialisation est que si une classe *A* est super-classe d'une classe *B*, alors toute instance de *B* est également une instance de *A*. La sémantique de cette relation est celle de l'inclusion ensembliste : les individus appartenant à l'interprétation d'une classe doivent aussi appartenir à celle de ses super-classes [84]. Le mécanisme de partage plaqué sur cette relation est l'héritage, qui est multiple en OWL 2.

Un axiome `SubClassOf` permet d'indiquer qu'une classe *B* est "une sorte de" *A*[4]. Si on déclare, en OWL 2, une classe *B* comme étant une spécialisation d'une classe *A*, cela n'implique pas le fait que *B* doive imposer des contraintes plus fortes sur les attributs définis pour *A*, ni qu'elle doive en ajouter de nouveaux. De plus, la classe *B* hérite toutes les propriétés de *A*, qu'elle peut spécialiser en spécifiant des restrictions de cardinalité ou d'appartenance sur les images des propriétés héritées.

AROM impose la spécialisation simple autant pour les classes que pour les associations, afin d'éviter les conflits de nommage inhérents à l'héritage multiple. Néanmoins, cette restriction a des répercussions importantes vis-à-vis du pouvoir d'expression du langage. Dans les conditions de l'héritage multiple, on pourrait construire, par exemple, des intersections entre des classes incomparables par la relation de spécialisation. Ces intersections n'ont pas de sens dans un scénario de mono-instanciation et d'héritage simple.

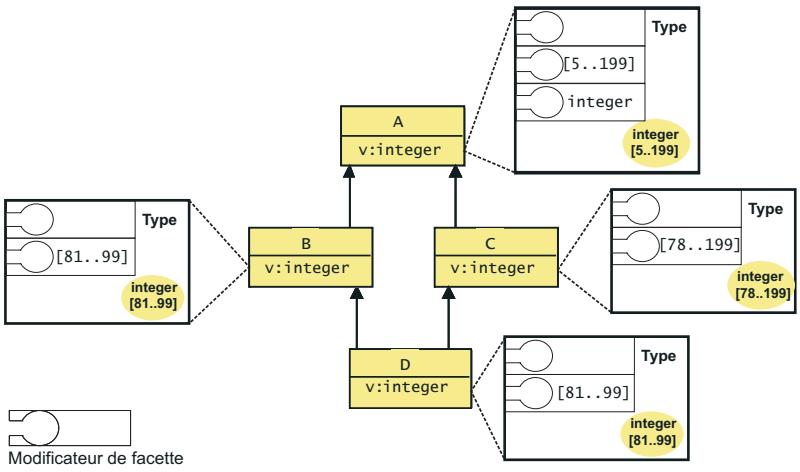


FIG. *5.14: Exemple d'héritage par D d'une variable v par deux chemins différents. La variable v de B étant plus spécifique, elle masque la variable v de C dans le processus de spécialisation.*

L'extension AROM-ONTO autorise l'héritage multiple, dans des conditions analogue à celles imposées par OWL 2 [168]. Concrètement, AROM-ONTO impose l'unicité de noms d'*attributs* à l'intérieur d'une base de connaissances. Cela veut dire que si une *classe A*, utilise la *variable v* pour modéliser un de ses *attributs*, la déclaration d'une nouvelle *variable* appelée *v* à l'intérieur d'une autre *classe* (ou *association*) est interdite. Cependant, la spécialisation d'une des *facettes* de *v* dans des *classes* qui héritent de *A* est permise. Par ailleurs, dans le cas où la variable *v* est héritée par deux chemins différents, on choisit pour la nouvelle classe la définition la plus spécifique de

[4] En réalité, cet axiome est beaucoup plus puissant, car il est défini pour des expressions à base de classes et pas seulement pour les classes simples. Cependant, pour illustrer nos propos par rapport à l'héritage multiple, nous avons choisi de faire cette simplification qui ne sera cependant pas poursuivie au déla de la section courante.

v qui masque toutes les autres. Dans la figure 5.14, *v* peut être héritée de *B* ou de *C* par *D*. La variable *v* de *B* étant plus spécifique, c'est elle qui sera retenue dans le processus de spécialisation.

Dans le cas où une relation d'inclusion ne peut pas être établie entre les domaines définis pour la variable *v* par les classes *B* et *C*, pour la classe *D* le domaine de *v* est déterminé comme l'intersection des deux domaines. Si le domaine de la variable *v* défini par la classe *B* et le domaine défini par la classe *C* pour la même variable sont disjoints, alors la base de connaissances est incohérente ; la classe *D* est équivalente à `Nothing`.

5.4.3 Différences secondaires de représentation entre AROM et OWL 2

Du point de vue de la représentation, les deux langages présentent aussi de nombreuses différences d'un ordre moins important que les précédentes.

Expressions à base de propriétés

Les propriétés de OWL 2 peuvent être utilisées pour définir des *expressions à base de propriétés*. Ainsi, pour les propriétés-lien, OWL 2 propose deux types d'expressions : `ObjectProperty`, la forme la plus simple d'une expression à base de propriétés-lien, et `InverseObjectProperty`, dont l'objectif est de permettre la navigation bidirectionnelle entre les expressions à base de classes (décrites dans la section suivante) définies comme *domaine* et *image* de la propriété considérée. Pour les attributs, OWL 2 offre un seul type d'expression, `DataProperty`, introduit pour des raisons d'homogénéité des spécifications.

En AROM, ce genre de définition implique l'existence d'associations capables de traiter les propriétés comme des individus, or le méta-modèle actuel d'AROM n'autorise pas de constructions similaires. Pour résoudre cette incompatibilité, et pour pouvoir réaliser des déclarations équivalentes en AROM-ONTO, nous avons défini un nouveau type d'association au niveau du méta-modèle d'AROM-ONTO, `AROMPropertyExpression`, spécialisée par `AROMProperty` et `AROMInverseProperty` (voir figure 5.15). Une instance de la méta-classe `AROMInverseProperty` définie par rapport à une propriété associe *P*, représente un ensemble de tuples $\langle x_i, y_i \rangle, i \geq 0$ tels que les tuples inverses $\langle y_i, x_i \rangle, i \geq 0$ appartiennent à l'extension de *P*. La figure 5.15 présente les extensions opérées au niveau de l'API du noyau AROM, afin de faciliter la définition des expressions à base de propriétés inverses, à l'instar de OWL 2.

Une instance de la méta classe `AROMInverseProperty` est une *propriété abstraite* (qui n'a pas de tuples propres), et qui contient seulement des liens vers des tuples *virtuels* déduits à partir de l'extension de sa propriété attachée, *P*. Ainsi, son extension est complètement dépendante de l'extension de *P*. Lors de la suppression d'un tuple $\langle x, y \rangle$ de l'extension de *P*, le lien vers le tuple virtuel $\langle y, x \rangle$ est supprimé de l'*extension* de la propriété inverse de *P*. Lors de l'ajout d'un tuple $\langle x, y \rangle$ à l'extension de *P*, un nouveau lien vers le tuple virtuel $\langle y, x \rangle$ est ajouté dans la propriété inverse de *P*. Les propriétés inverses ne spécialisent aucune propriété. En revanche, elles peuvent être spécialisées par tout type d'expression à base de propriétés.

Si la propriété inverse d'une propriété *P* est spécialisée par une propriété *Q*, alors la propriété *P* sera également spécialisée par la propriété inverse de *Q*, afin de garantir la cohérence des extensions des quatre expressions à base de propriétés.

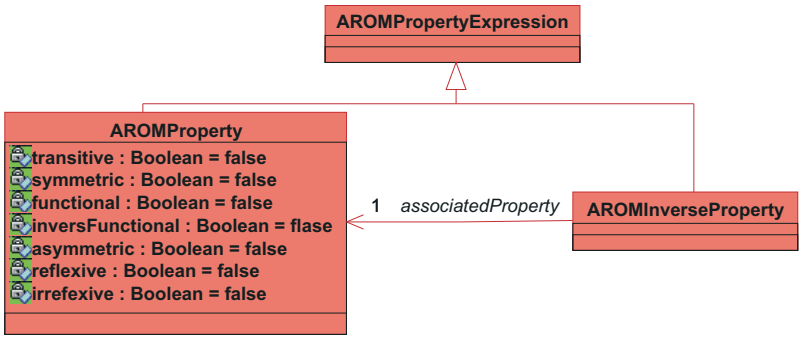


FIG. *5.15: Diagramme de classes UML qui définit la structure des expressions à base de propriétés en AROM.*

Cette extension du méta-modèle est complétée par une extension du langage de description des bases de connaissances AROM-ONTO (voir annexe A), avec une primitive qui permet de désigner de façon générique l'inverse d'une propriété. La sémantique de AROM-ONTO a été également étendue pour prendre en compte cette nouvelle structure comme illustré dans l'annexe B.

Expressions à base de classes

Au niveau des classes, OWL 2 offre une plus large palette d'options et de possibilités de modélisation que ne le fait AROM. OWL 2 permet de définir et d'utiliser des *expressions à base de classes* (appelées *concepts complexes* en Logiques de Description) comme super-classes d'une classe donnée, et/ou domaine ou image d'une expression à base de propriétés, *etc.* On peut voir les *expressions à base de classes* comme des classes abstraites[5]. Elles offrent un cadre de transit, pour rendre les définitions de classe dont elles sont les super-classes plus simples et faciles à comprendre. Une *expression à base de classes* est interprétée comme un ensemble d'individus. Concrètement, sa définition consiste à spécifier un ensemble de conditions d'appartenance. Les individus qui respectent ces conditions sont considérés comme des instances de *l'expression à base de classes* correspondante.

Comme illustré dans la figure 5.16, OWL 2 met à disposition un ensemble riche de primitives qui peuvent être utilisées pour la construction des expressions à base de classes. Il s'agit notamment de :

- connecteurs booléens `and`, `or` et `not` appliqués aux extensions de classe, opérations modélisés par les méta-classes `ObjectIntersectionOf`, `ObjectUnionOf` et respectivement `ObjectComplementOf` [181],
- une forme restreinte du quantificateur universel (`only`) implémenté à travers la méta-classe `ObjectAllValuesFrom` [181],
- quantificateur existentiel (`some`) implémenté par `ObjectSomeValuesFrom` [181],
- restrictions de cardinalité implémentées à l'aide des méta-classes `ObjectExactCardinality`, `ObjectMinCardinality` et `ObjectMaxCardinality` [181],
- la primitive `ObjectOneOf`, qui décrit une classe en énumérant les individus qui en font partie (son extension),
- une forme restreinte de la restriction `self` (`ObjectHasSelf`) [181].

[5] Dans cette thèse, nous utilisons le terme *classe abstraite* pour dénoter une classe qui ne détient pas d'instances propres, mais qui contient des liens vers les instances d'autres classes.

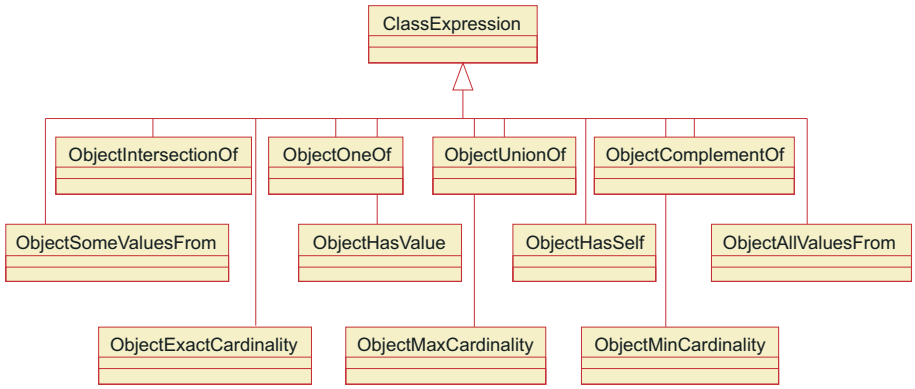


FIG. *5.16: Diagramme de classes UML qui définit la structure des expressions à base de classes de OWL 2. Source [181]*

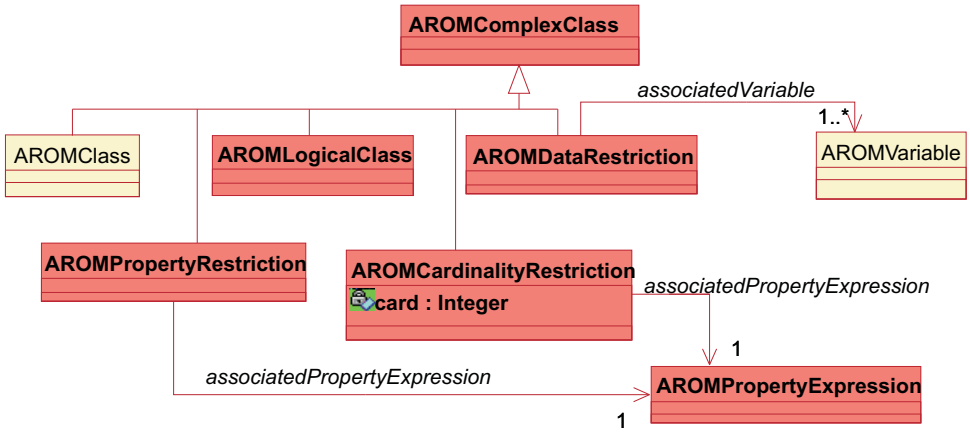


FIG. *5.17: Extrait de la hiérarchie de classes du méta-modèle AROM-ONTO.*

AROM ne permet pas des définitions de ce type. Son langage de description de bases de connaissances ne dispose pas de primitives nécessaires pour définir des classes complexes. Néanmoins, pour l'extension AROM-ONTO, nous avons adopté une approche similaire à celle de OWL 2 concernant la définition des différentes types d'expression à base de classes, présentées dans [181] comme des spécialisations de la classe générique `ClassExpression`. Nous proposons alors d'introduire dans le méta-modèle d'AROM-ONTO, un nouveau type de structure de représentation `AROMComplexClass`, qui modélise les caractéristiques générales des expressions à base de classes (voir figure 5.17). La méta-classe `AROMClass` est considérée à la fois comme une structure de représentation élémentaire (sous-classe de `AROMStructure`) et comme expression à base de classes (sous-classe de `AROMComplexClass`). Elle est la seule méta-classe dont les instances représentent des classes *concrètes*. La méta-classe `AROMComplexClass` dispose en plus, de quatre autres spécialisations qui définissent des classes *abstraites*[6] :

- `AROMLogicalClass` pour la définition des opérations ensemblistes entre classes,
- `AROMRestriction` qui modélise les restrictions de propriétés. Celles-ci sont définies pour une *expression à base de propriétés* pour laquelle elles imposent l'expression à base de classe désignée par le rôle `range`.

[6]Une classe *abstraite* est une classe qui ne dispose pas d'instances propres.

– `AROMCardRestriction` pour les restrictions de propriétés définies par rapport à un nombre minimum/maximum d'occurrences d'une *expression à base de propriétés*. Ces restrictions peuvent aussi imposer une expression à base de classes pour le rôle `range` de l'expression à base de propriétés visée, mais cela n'est pas obligatoire.
– `AROMDataRestriction` qui modélise des restrictions sur des variables, définies par rapport à un domaine donné de type intervalle, ou ensemble, ou par rapport à une valeur donnée.

Dans la suite, nous décrivons la structure et le comportement de chaque type d'expression à base de classes.

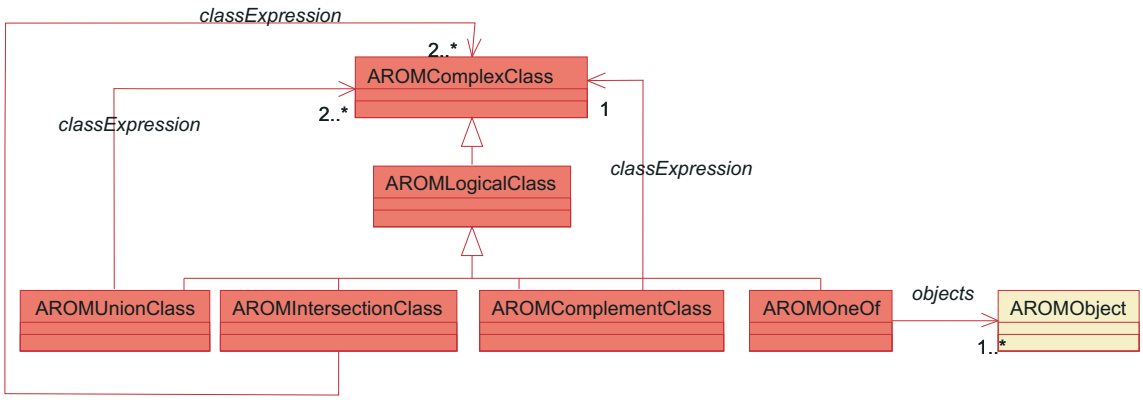


FIG. *5.18: Extrait de la hiérarchie de classes du méta-modèle AROM-ONTO.*

AROMLogicalClass

La figure 5.18 illustre la structure des expressions à base de classes construites en utilisant les opérateurs classiques pour les ensembles, appelés *conjonction* (`and`), *disjonction* (`or`) et *négation* (`not`) dans les langages logiques, appliquées à des extensions de classe et en utilisant la description explicite d'un ensemble d'individus (modélisée par la méta-classe `AROMOneOf`). Ainsi, la méta-classe générique `AROMLogicalClass` dispose de quatre spécialisations (voir figure 5.18), correspondant à chacun des quatre opérateurs évoqués auparavant :

– la méta-classe `AROMIntersectionClass` permet de modéliser l'ensemble des individus qui sont attachées à chacune des expressions à base de classes spécifiées comme paramètres. Dans le langage de description des bases de connaissances, une expression d'intersection de classes est de la forme : `intersection-of(`CE_1, ..., CE_n`)`;
– `AROMUnionClass` modélise l'ensemble des individus qui sont attachés à au moins une des expressions à base de classes spécifiées comme paramètre de l'union. Une expression d'union de classes est de la forme : `union-of(`CE_1, ..., CE_n`)`;
– une expression à base de classes de type complément (instance de `AROMComplementClass`) contient les individus qui ne sont pas instances de l'expression à base de classes spécifiée comme paramètre. Par exemple, l'expression `complement-of(CE)`, représente l'ensemble des individus qui ne sont pas attachés à l'expression à base de classes `CE`.
– une expression à base de classe de type énumération d'individus, modélisée par la méta-classe `AROMOneOf`, contient exactement les individus spécifiées comme paramètres. Par exemple, l'énumération d'individus `one-of{`o_1, o_2, o_3`}` contient exclusivement les individus o_1, o_2 et o_3.

Dans la suite, nous présentons la construction et le comportement de la classe d'intersection, `AROMIntersectionClass` sur la base de l'exemple de la figure 5.19. Les détails sur la structure et le comportement des classes `AROMUnionClass`, `AROMComplementClass` et `AROMOneOf` peuvent être consultés dans [168].

Étant donnée la hiérarchie de classes de la figure 5.19, on cherche à trouver l'intersection des classes *B* et *C*, notée `intersection-of(B,C)`. L'extension de *B* contient les instances propres à *B* mais aussi toutes les instances propres de *G*, *I*, *H* et *J* qui sont ses spécialisations. En suivant les liens de spécialisation, on observe que *B* et *C* partagent des instances au niveau des sous-classes *J* et *H* puisque celles-ci ont comme super-classes indirectes communes *B* et *C*. L'intersection de *B* et *C* est une expression à base de classes, de type `AROMIntersectionClass`.

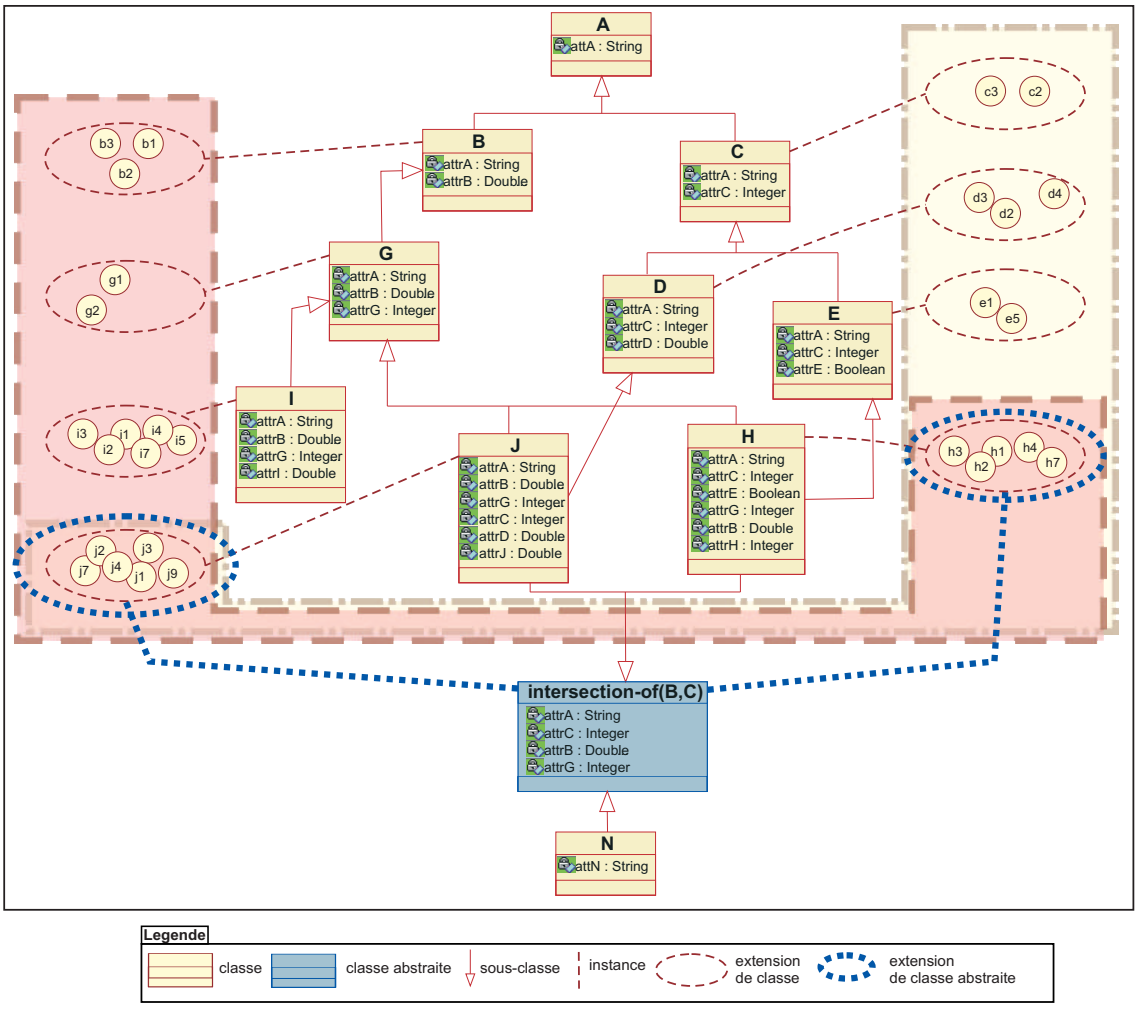


FIG. *5.19: Intersection de deux classes (B et C) partageant deux sous-classes (J et H)*

`AROMIntersectionClass` est une méta-classe qui permet de construire une classe comme l'intersection de deux ou plusieurs classes opérandes. La classe résultante est une généralisation de toutes les classes impliquées dans sa définition (qui ont des instances dans l'intersection, dans l'exemple choisi les classes *J* et *H*) et possède, en conséquence une *intension*. En revanche, elle n'a pas d'instances propres mais le système maintient des liaisons vers les instances de ses sous-classes (j_1, j_2, j_3, j_4, j_7, j_9 et h_1, h_2, h_3, h_4, h_7). Comme toute autre classe, elle peut à son tour

être spécialisée par n'importe quelle classe (définie ou de base).

Si une classe *N* hérite d'une classe d'intersection `intersection-of(B,C)`, elle hérite de l'*intension* la plus générale de toutes les classes qui ont des instances dans l'intersection, elle peut affiner cette *intension* et posséder des instances propres. Une classe d'intersection (*i.e.* `intersection-of(B,C)`) dépend de ses sous-classes (*J* et *H*). Dès lors qu'une des classes d'un des chemins de spécialisation perd une instance (*i.e.* par suppression) la classe d'intersection la perd aussi. Si nous supprimons l'instance j_2 de *J*, elle sera aussi supprimée de `intersection-of(B,C)`. Le même type de mise à jour dynamique est réalisé en cas d'ajout d'instance.

Pour des raisons de cohérence, au moment de l'ajout d'une nouvelle classe *N* dans le modèle, le système vérifie si *B* et *C* sont des ancêtres de *N* et si c'est le cas, il ajoute les liaisons nécessaires pour que la classe d'intersection `intersection-of(B,C)` reflète le changement. `intersection-of(B,C)` sera, en conséquence, super-classe de *N*, et aura des liaisons vers toutes ses instances (voir figure 5.19).

Au niveau du langage de description des bases de connaissances AROM-ONTO, nous avons introduit de nouvelles primitives : `intersection-of`, `union-of`, `complement-of` et `one-of`. Un exemple d'utilisation de la primitive `union-of` est donné dans la figure 5.20. La définition BNF de ces nouvelles primitives est présentée dans l'annexe A. Nous avons également étendu la fonction d'interprétation d'une base de connaissances afin qu'elle prenne en compte les nouvelles constructions (voir annexe B).

AROM-ONTO

```
class : HolidayDestination
super-class: union-of(SeasideRegion, MountainRegion)
```

FIG. *5.20: Exemple d'utilisation d'une classe complexe en AROM-ONTO.*

AROMPropertyRestriction

Afin de manipuler une restriction de propriété en AROM-ONTO, nous avons introduit une nouvelle méta-classe dans son méta-modèle : `AROMPropertyRestriction`, une spécialisation de `AROMComplexClass` (voir figure 5.21). Pour traiter de façon adaptée les quatre types de restriction applicables aux propriétés OWL 2, la classe `AROMPropertyRestriction` est spécialisée en fonction de la contrainte désirée. Elle dispose, en conséquence, de quatre sous-classes : `AROMSomeRestriction`, `AROMAllRestriction`, `AROMHasValueRestriction` et `AROMSelfRestriction`.

Chaque restriction de propriétés en OWL 2 est définie par rapport à une expression à base de propriétés, que nous appelons *propriété associée*. Ce lien est défini au niveau du méta-modèle AROM-ONTO par la relation de dépendance qui existe entre la méta-classe `AROMPropertyRestriction` et la classe `AROMPropertyExpression` (voir figure 5.21), héritée par les quatre spécialisations de `AROMPropertyRestriction`.

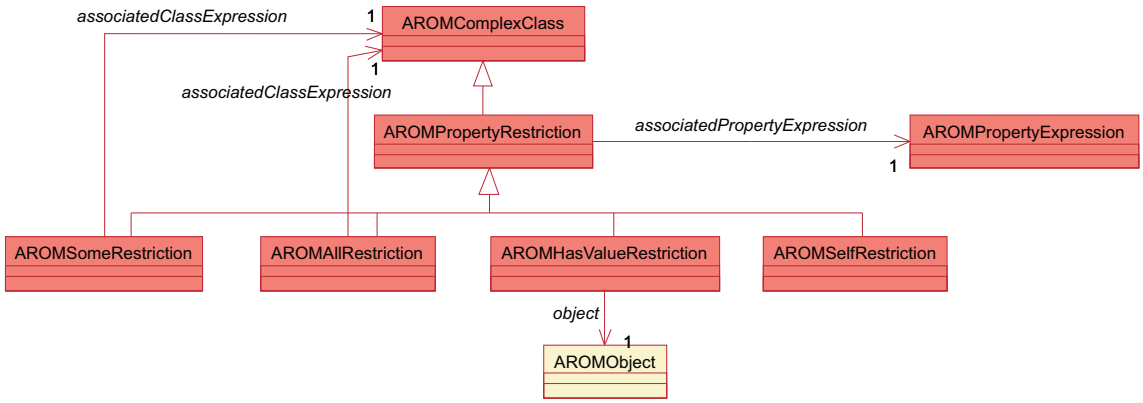


FIG. *5.21: La hiérarchie de méta-classes utilisée pour définir les restrictions de propriété en AROM-ONTO.*

Une restriction de propriété AROM-ONTO est une classe abstraite, qui peut être vue comme un filtre, utile pour sélectionner les individus du domaine de la *propriété associée* qui satisfont une contrainte spécifique au type de restriction employé. Par exemple, la restriction `AROMAllRestriction` fait référence à l'ensemble des individus pour lesquels toutes les valeurs de la *propriété associée* sont dans un ensemble donné, spécifié à travers une expression à base de classes. Le schéma générique d'une restriction est présenté par la figure 5.22. Ainsi, une restriction peut être vue comme une classe abstraite, dont l'extension est incluse dans l'extension du domaine de la propriété P. Elle est impliquée dans une propriété *abstraite* P' qui hérite de la sémantique de P, mais qui s'applique à une image restreinte B'. Le domaine A, ainsi que les images B et B', peuvent être spécifiés à travers des expressions à base de classes. Il n'y a pas de contraintes sur l'intension et/ou l'extension de A, B et B'.

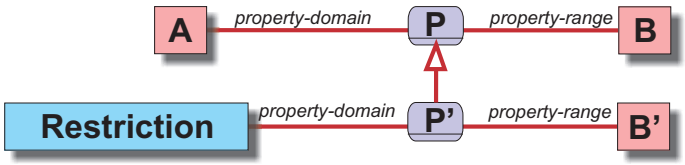


FIG. *5.22: Le schéma général des liens dans lesquels est impliquée une restriction de propriété.*

La figure 5.23 montre un exemple concret de restriction de type `AROMSomeRestriction` décrivant l'ensemble des individus (instances de la classe `Person`) qui ont au moins un chien comme animal de compagnie. L'extension de la restriction `some hasPet Dog` est composée d'un ensemble de liens vers les objets du domaine initial (instances de la classe `Person`) qui respectent les conditions imposées.

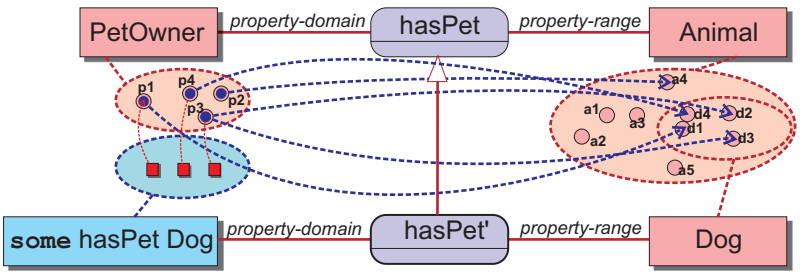


FIG. *5.23: Exemple de restriction existentielle qui décrit la classe d'individus qui possèdent au moins un chien comme animal de compagnie.*

Une restriction existentielle est complètement dépendante de sa *propriété associée*, ainsi que du domaine et de l'image de cette dernière. Pour une meilleure compréhension, nous illustrons ces dépendances sur l'exemple de la figure 5.23. Au moment de la suppression d'une instance de la classe `Person` qui satisfait la propriété `hasPet'`, comme par exemple p_4, l'élément correspondant de l'extension de `some hasPet Dog` sera également supprimé. Cependant, lors de la modification de l'instance p_4, qui n'entraîne pas la modification des tuples dans lesquels p_4 est impliquée, l'extension de `some hasPet Dog` reste la même. L'adjonction d'une nouvelle instance à `Person`, qui est impliquée dans une relation `hasPet'` a comme résultat la création d'un nouveau lien depuis `some hasPet Dog` vers ce nouvel objet.

La suppression d'un tuple appartenant à `hasPet'` (dans notre exemple $\langle p_1, d_1 \rangle$), sans lequel un élément de `Person` n'est plus lié à une instance de `Dog`, suppose la suppression de l'élément correspondant de `some hasPet Dog`. Néanmoins, si on supprime le tuple $\langle p_3, d_3 \rangle$, p_3 a toujours un lien de type `hasPet` avec le chien d_2 et, en conséquence il reste dans l'extension de `some hasPet Dog`.

Lorsqu'on supprime une instance de `Dog`, on supprime aussi tous les tuples dans lesquels elle a été impliquée. Les instances du domaine qui n'auront plus de lien de type `hasPet` avec un chien seront aussi éliminées depuis l'extension de `some hasPet Dog`. Une restriction existentielle peut être spécialisée. Les instances de ses sous-classes doivent avoir des relations `hasPet'` avec au moins un individu appartenant à la restriction d'image `Dog`.

La restriction `AROMAllRestriction` est construite de façon analogue, avec la différence que toutes les objets référencés par la classe de restriction (`Restriction`) doivent être reliés par la *propriété associée* P' exclusivement à des objets attachés à la classe restreinte B'. La restriction `AROMSelfRestriction` désigne les objets qui sont reliés par la *propriété associée* à eux-mêmes, et la restriction `AROMHasValueRestriction` spécifie à la place d'une expression de classe `B'` (voir figure 5.22), un seul individu. Pour plus de détails sur la construction de ces classes, le lecteur peut consulter [168, 167].

Pour soutenir la définition des restrictions de propriétés à travers le langage de description de bases de connaissances AROM-ONTO, nous avons défini les primitives `some`, `all`, `has-value` et `has-self`. Deux exemples d'utilisation de la restriction `some` sont donnés dans la figure 5.24. Le premier vise la définition de la classe `DogOwner` comme sous-classe de la classe des individus qui ont au moins une relation `hasPet` avec une instance de `Dog`. Pour définir la classe des chiens appartenant à des enfants (`DogWithChildOwner`) on peut utiliser l'expression à base de propriétés `inverse-of(hasPet)` comme illustré dans le deuxième exemple (voir figure 5.24).

```
AROM-ONTO
property: hasPet
 property-domain
    type: PetOwner
 property-range
    type: Animal

class:DogOwner
 super-class: Person, some(hasPet Dog)

class:Child
 super-class: Person

class:DogWithChildOwner
 super-class: some(inverse-of(hasPet) Child)
```

FIG. *5.24: Exemple de description AROM-ONTO qui utilise une restriction de propriété.*

La définition BNF des nouvelles primitives est donnée dans l'annexe A. Nous avons également étendu la fonction d'interprétation d'une base de connaissances afin qu'elle prenne en compte les nouvelles constructions (voir annexe B).

AROMCardinalityRestriction

En OWL 2, on peut décrire une classe comme étant l'ensemble des individus ayant *au moins*, *au plus*, ou *exactement card* relations d'un type donné avec d'autres individus. Pour pouvoir réaliser des déclarations analogues en AROM-ONTO, nous avons ajouté à son méta-modèle la classe `AROMCardinalityRestriction`, spécialisée par `AROMMinCardinality`, `AROMMaxCardinality` et `AROMExactCardinality`, comme illustré dans la figure 5.25. Une instance de la classe `AROMMinCardinality` est une restriction qui impose pour ses objets l'existence d'au moins *card* relations d'un certain type, *PE*, précisé par l'*expression à base de propriété associée* (voir figure 5.25), avec des individus attachés à l'*expression à base de classe associée*, *CE*. Si la restriction n'est pas définie par rapport à une *expression à base de classes* (*i.e. CE* n'est pas précisée), alors toutes les instances attachées à *l'expression à base de propriété PE* seront prises en compte pour le calcul de la cardinalité. Les restrictions de cardinalité étant construites de façon très similaire aux restrictions de propriétés (`AROMPropertyRestriction`), nous ne reprenons pas ici les détails de leur construction. Ces méta-classes sont également présentées dans [168].

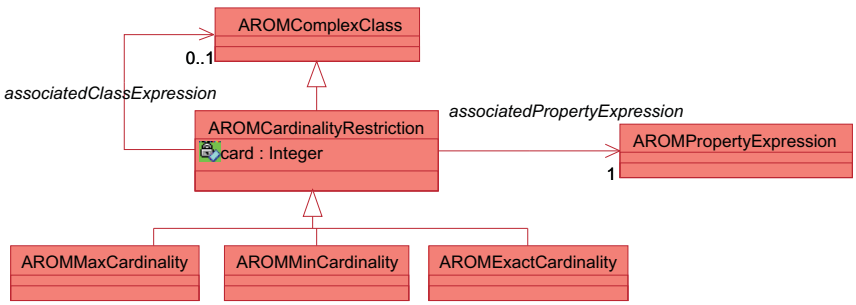


FIG. *5.25: La hiérarchie de méta-classes utilisée pour définir les restrictions de cardinalité en AROM-ONTO.*

Nous avons étendu le langage de description des bases de connaissance AROM-ONTO avec des primitives qui facilitent la définition des restrictions de cardinalité, comme illustré dans l'an-

nexe A. Un exemple de définition d'une restriction de cardinalité est donné dans la figure 5.26. La fonction d'interprétation d'une base de connaissances AROM-ONTO a été également étendue pour prendre en compte les nouvelles primitives (voir annexe B).

```
AROM-ONTO
property: hasHouse
 property-domain
       type: Person
 property-domain
       type: Address

class:HouseOwner
 super-class: Person, min(1 hasHouse)
```

FIG. *5.26: Exemple de définition en AROM-ONTO d'une restriction de cardinalité.*

Restriction sur les variables

En OWL 2, on peut créer des expressions à base de classes en imposant des restrictions sur les propriétés à valeurs concrètes, qui fonctionnent de façon similaire aux restrictions sur les propriétés-lien. Ainsi, OWL 2 propose deux types de restrictions pour les attributs : *(i)* les restrictions construites à l'aide des opérateurs booléens `same`, `all` et à l'aide de l'énumération des valeurs et *(ii)* les restrictions de cardinalité. Nous rappelons que l'équivalent en AROM des attributs de OWL 2 sont les variables.

AROMDataRestriction

La figure 5.27 montre les méta-classes de AROM-ONTO utilisées pour définir des restrictions sur les valeurs des variables, à l'instar des restrictions sur les attributs de OWL 2. La classe générique `AROMDataRestriction` modélise les caractéristiques communes aux restrictions considérées. Elle a une association avec la méta-classe `AROMVariable`, héritée par ses deux spécialisations : `AROMSomeValueIn` et `AROMAllValuesIn`. La relation `associatedVariable` est redéfinie pour la restriction `AROMValueEquals`, car celle-ci est spécifiée par rapport à une seule *variable associée* (voir figure 5.27).

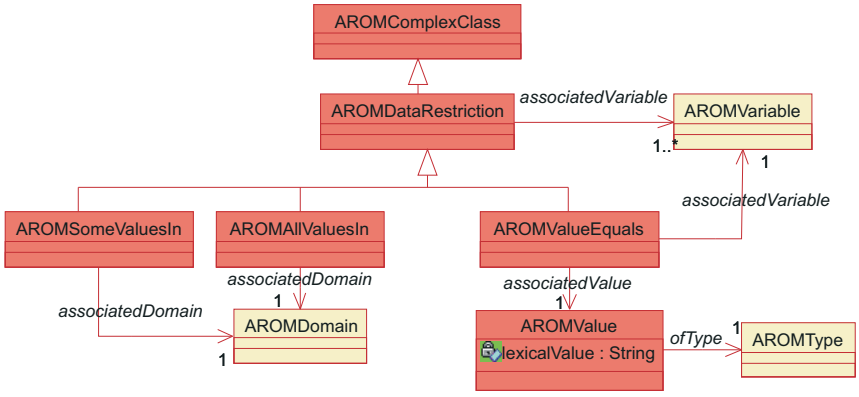


FIG. *5.27: La hiérarchie de méta-classes utilisée pour définir les restrictions sur les variables en AROM-ONTO.*

Les restrictions de type `AROMSomeValuesIn` et `AROMAllValuesIn` sont spécifiées par l'inter-

médiaire d'un ensemble de *variables associées*, $v_1, v_2, \ldots, v_n, n \geq 1$, auxquelles s'ajoute un *domaine associé*, *D*. Les variables $v_1, v_2, \ldots, v_n$ doivent avoir le même type. La restriction existentielle modélisée par la méta-classe `AROMSomeValuesIn` permet de définir une classe comme étant l'ensemble des individus pour lesquels les valeurs des *variables associées* sont *précisées* et appartiennent au domaine *D*. La restriction universelle définie par la méta-classe `AROMAllValuesIn`, modélise une classe comme étant un ensemble d'individus, pour lesquels les valeurs des variables $v_1, v_2, \ldots, v_n$, *si elles sont connues*, sont dans le domaine *D*. La restriction `AROMValueEquals` est définie par rapport à une valeur (instance de `AROMValue`) ayant comme type l'un des types définis par AROM (`AROMType`).

Une restriction existentielle sur un ensemble de variables $v_1, v_2, \ldots, v_n$, est une classe abstraite C_{ER} qui contient des liens vers les instances de la classe la plus générique qui définit l'ensemble de variables $v_1, v_2, \ldots, v_n$. Comme les individus recherchés doivent avoir des valeurs pour chaque variable $v_i, i \in 1..n$, et parce que la multi-instantiation n'est pas autorisée en AROM-ONTO, les définitions des variables $v_1, v_2, \ldots, v_n$ doivent être regroupées dans une même classe. Ainsi, la construction d'une restriction existentielle sur un ensemble de variables, consiste, dans un premier temps, à déterminer la classe *C* qui définit toutes les variables $v_1, v_2, \ldots, v_n$. Si une telle classe n'existe pas dans la fermeture transitive de la base de connaissances considérée, l'extension de C_{ER} est vide. Dans le cas contraire, depuis l'ensemble d'instances de *C*, le système extrait un sous-ensemble maximal, pour lequel chaque variable v_i a une valeur et cette valeur est incluse dans le domaine *D*. L'extension de la classe de restriction C_{ER} contient des liens vers chaque instance incluse par le sous-ensemble de *C* obtenu. La figure 5.28 illustre l'utilisation d'une restriction existentielle sur la variable `age` pour la définition de la classe `Adult`.

AROM-ONTO

```
class: Person
variables:
 variable: name
    type: string
 variable: age
    type: integer
    domain: [0..150]

class: Adult
super-class: Person, some-values-in(age [18..150])
```

FIG. *5.28: Exemple de description AROM-ONTO qui utilise une restriction de variable.*

Une restriction universelle sur l'ensemble de variables $v_1, v_2, \ldots, v_n$, est une classe abstraite C_{UR} qui contient des liens vers toutes les instances $x_k, k \geq 0$ de la base de connaissance qui :

- appartiennent à des classes pour lesquelles les variables $v_i, i \in 1..n$ ne sont pas définies ;
- appartiennent à des classes pour lesquelles toutes ou une partie des variables $v_i, i \in 1..n$ sont définies, et pour chaque $x_j, j \in 1..k$ la valeur de chaque variable $v_i, i \in 1..n$, *si elle est connue*, est dans le domaine *D*.

Comme illustré dans la figure 5.6, deux types de domaines sont pris en compte par le méta-modèle de AROM-ONTO, et définis comme des restrictions de l'un des types reconnus par le système (pour une présentation des types utilisés par AROM-ONTO voir section 6.3.1). Le premier type de domaine, `AROMSetDomain`, permet de définir des domaines concrets par le biais de l'énumération explicite de l'ensemble de valeurs contenues dans ces domaines. Les domaines de type intervalle, modélisés par la méta-classe `AROMIntervalDomain`, sont définis par rapport à

un type ordonné en précisant deux valeurs extrêmes (le `minimum` et le `maximum`) qui limitent les valeurs acceptées par ces domaines. Un domaine de type intervalle, peut avoir comme type associé, hormis des types numériques, le type `string`. Un exemple d'intervalle de type `string` est `['f'..'xyz']`. Celui-ci contient toutes les chaînes de caractères de longueur minimum 1 et maximum 3 et qui sont *supérieurs* à 'f' (*i.e.* 'g', 'h', ..., 'aa', 'ab', ...) et *inférieurs* à 'xyz' (*i.e.* 'xyy', 'xab', ..., 'aaa', ..., 'b', 'a').

Nous avons étendu le langage de description des bases de connaissances AROM-ONTO avec les primitives `some-values-in`, `all-values-in` et `value-is`, afin de rendre possible la définition des restrictions sur les variables. Leur définition BNF est donnée dans l'annexe A. Nous avons également étendu la fonction d'interprétation d'une base de connaissances afin qu'elle prenne en compte les nouvelles primitives (voir annexe B).

AROMDataCardinality

Comme illustré dans la figure 5.29, les restrictions de cardinalité sur les variables AROM-ONTO sont exclusivement définies pour les variables multivaluées (dont le type est `List` ou `Set`). Les restrictions de cardinalité peuvent être *qualifiées* ou *non-qualifiées*. Dans le premier cas, les valeurs prises en compte par la restriction de cardinalité appartiennent à un domaine donné (le domaine associé). Dans le deuxième cas, la restriction vise l'ensemble de valeurs de la variable associée. Une restriction de cardinalité est une classe abstraite qui désigne l'ensemble des individus pour lesquels la variable associée contient respectivement *minimum*, *maximum* et *exactement card* valeurs, appartenant au domaine associé à la restriction, si celui-ci est spécifié.

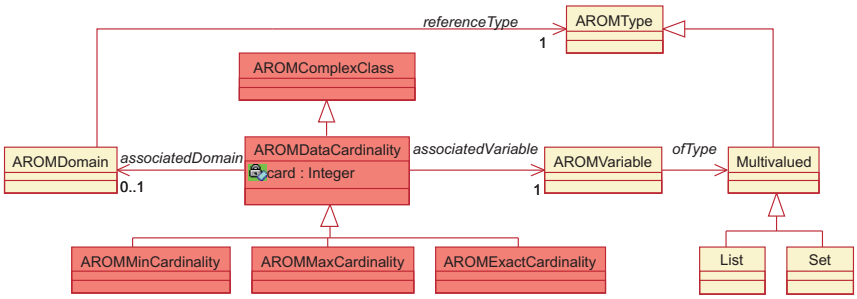


FIG. *5.29: La hiérarchie de méta-classes utilisées pour définir, en AROM-ONTO, les restrictions de valeur sur les variables.*

Par exemple, une restriction de cardinalité non-qualifiée, sur la variable `nom` de la base de connaissances illustrée dans la figure 5.30, est définie en AROM-ONTO comme suit : `variable-min-card (nom 2)`. Cette restriction a du sens car la variable `nom` est multi-valuée, ayant comme type associé le type *liste de chaînes de caractères* (`list-of string`). La restriction `variable-min-card (nom 2)` représente l'ensemble d'individus $\{p_1, p_2\}$. L'individu p_3 n'appartient pas à la restriction de cardinalité, car la valeur de la variable `nom` lui étant attachée contient un seul élément.

```
AROM-ONTO
  class: Personne
  variables:
    variable: nom
      type: list-of string
  cardinality:
      min:1
      max: *

  instance: p1
  is-a: Personne
     nom = ["Dia Miron","Alina Miron","Alina Dia Miron"]

  instance: p2
  is-a: Personne
     nom = ["Aurelia Roman","Aura Roman"]

  instance: p3
  is-a: Personne
     nom = ["Radu Ioan"]
```

FIG. *5.30: La définition d'une variable multi-valuée en AROM-ONTO.*

Une restriction qualifiée pour la même variable `nom` est, par exemple, la restriction

```
variable-max-card (nom 1 {"Alina Miron", "Aura Roman", "Aurelia Roman"}),
```

qui représente l'ensemble d'individus $\{p_1, p_3\}$. L'individu p_2 n'est pas pris en compte car, pour la propriété `nom`, il dispose de deux valeurs, les deux dans le domaine attaché à la restriction. Afin de pouvoir définir des restrictions de cardinalité pour les variables multi-valuées, nous avons étendu le langage de description des bases de connaissances avec des nouvelles primitives (voir annexe A). L'extension de la fonction d'interprétation d'une base de connaissances est présentée dans l'annexe B.

Axiomes

Les éléments principaux d'une ontologie OWL 2 sont les axiomes, des déclarations qui sont considérées vraies dans un domaine donné. OWL 2 offre une large palette d'axiomes, construites, dans la spécification structurelle, comme des spécialisations de la méta-classe `Axiom` (voir figure 5.31). Il s'agit notamment de déclarations, d'axiomes de classe, d'axiomes de propriété, de définition de types de données, de faits, *etc.* Les axiomes sont des structures de représentation à part entière, définies par rapport à des entités et/ou des expressions OWL 2.

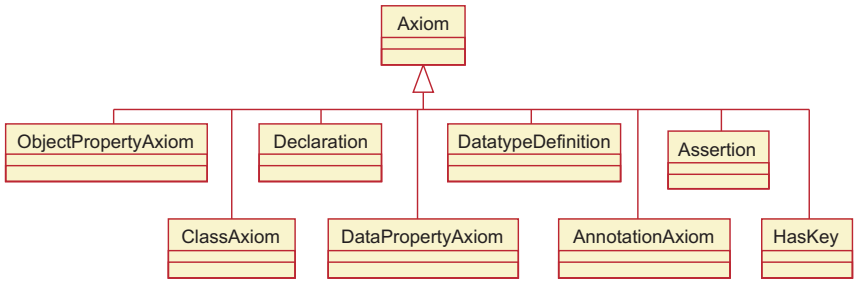


FIG. *5.31: Le méta-modèle utilisé pour définir les axiomes en OWL 2.*

Axiomes de spécialisation.

Les expressions à base de classes de OWL 2 sont organisées dans une hiérarchie de spécialisation, en utilisant des axiomes du type `SubClassOf`, définis pour des paires d'expressions à base de classes, comme, par exemple, `SubClassOf(`$\texttt{CE}_1$ $\texttt{CE}_2$`)`. L'interprétation d'une telle déclaration en OWL 2 est que l'expression $\texttt{CE}_2$ est plus générale que (subsume) l'expression $\texttt{CE}_1$. En suivant le même principe, OWL 2 facilite la définition des hiérarchies de spécialisation pour les expressions à base de propriétés-lien, à travers l'axiome `SubObjectPropertyOf`, et, pour les attributs, en utilisant l'axiome `SubDataPropertyOf`.

Le langage AROM n'offre pas la possibilité de définir de tels axiomes indépendants. Seuls deux types d'axiomes sont pris en compte par AROM : l'axiome de *sous-classe* et celui de *sous-association*, qui sont intégrés respectivement dans la structure de classe et dans celle d'association. Au niveau du langage de description de bases de connaissances AROM, les deux axiomes sont spécifiés par l'intermédiaire des primitives `super-class` et `super-association` (voir la spécification BNF du langage AROM-ONTO en annexe A).

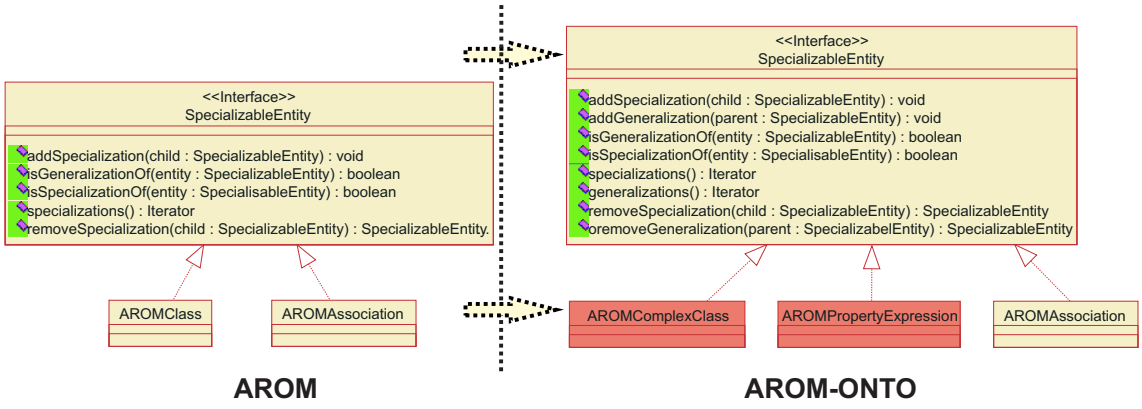


FIG. *5.32: La spécialisation en AROM et en AROM-ONTO.*

Comme illustré pas la figure 5.32, dans le méta-modèle AROM, la spécialisation est mise en place à travers l'interface `SpecializableEntity`, implémentée autant par les classes (`AROMClass`) que par les associations (`AROMAssociation`). Celle-ci offre des fonctionnalités pour l'ajout et la suppression de spécialisations pour la structure courante (classe ou association) et, dans sa version originale, gère une seule généralisation pour chaque structure. Pour l'extension AROM-ONTO, nous avons choisi une réorganisation du méta-modèle afin de permettre la définition de hiérarchies de multi-spécialisation pour les expressions à base de classes aussi bien que pour les associations de AROM-ONTO. Cette restructuration est illustrée dans la figure 5.32. Dans un premier temps, nous avons complété les spécifications de `SpecializableEntity` afin de permettre la gestion de plusieurs parents pour chaque structure de représentation (l'héritage multiple). Ensuite, nous avons défini comme entités spécialisables, à part les associations, les expressions à base de classes et les expressions à base de propriétés (voir figure 5.32). Nous avons modifié en conséquence le langage de description de bases de connaissances AROM-ONTO (voir annexe A) et sa sémantique formelle (voir annexe B).

Axiomes d'équivalence et de disjonction

Deux ou plusieurs expressions à base de classes peuvent être définies en OWL 2, comme étant *équivalentes* ou *disjointes*. Ce type de déclaration est réalisé en utilisant les axiomes `EquivalentClasses` et respectivement `DisjointClasses`. Un axiome d'équivalence indique que les expressions à base de classes impliquées dans l'axiome ont la même extension (décrivent exactement le même ensemble d'individus). Néanmoins, cela ne veut pas dire que ces expressions à base de classes ont la même signification intentionnelle (*i.e.* elles peuvent ne pas représenter le même concept). Un axiome de disjonction, défini entre deux ou plusieurs expressions à base de classes, impose une intersection vide entre les extensions des expressions à base de classes impliquées.

AROM n'offre pas la possibilité de définir des axiomes d'équivalence et de disjonction entre les différentes structures qu'il met à disposition. À première vue, on pourrait penser à modéliser ces axiomes comme des associations spéciales, ayant une sémantique et un comportement prédéfinies, dont les rôles désignent les structures qu'on veut définir comme étant équivalentes ou disjointes. Cependant, cela revient à modifier la définition même d'une association AROM, pour qu'elle permette de relier un nombre variable d'éléments (on ne connaît pas *a priori* la cardinalité d'une telle association). De plus, les rôles des associations AROM sont construits pour désigner des objets, et non pas des structures de représentation telles que les expressions à base de classes, les expressions à base de propriétés ou les variables.

Ainsi, nous avons imaginé les axiomes d'équivalence et de disjonction entre classes comme des éléments de représentation à part entière, introduits dans le méta-modèle de AROM-ONTO, à travers la méta-classe `AROMPredefinedClassAxioms` (voir figure 5.33). Celle-ci regroupe les caractéristiques de base d'un axiome à base de classes, telles que l'ensemble d'expressions à base de classes impliquées dans l'axiome, et dispose de deux spécialisations `AROMEquivalentClasses` et `AROMDisjointClasses`, qui modélisent respectivement la sémantique et les inférences liés à une déclaration d'équivalence et à une déclaration de disjonction entre plusieurs expressions à base de classes associées.

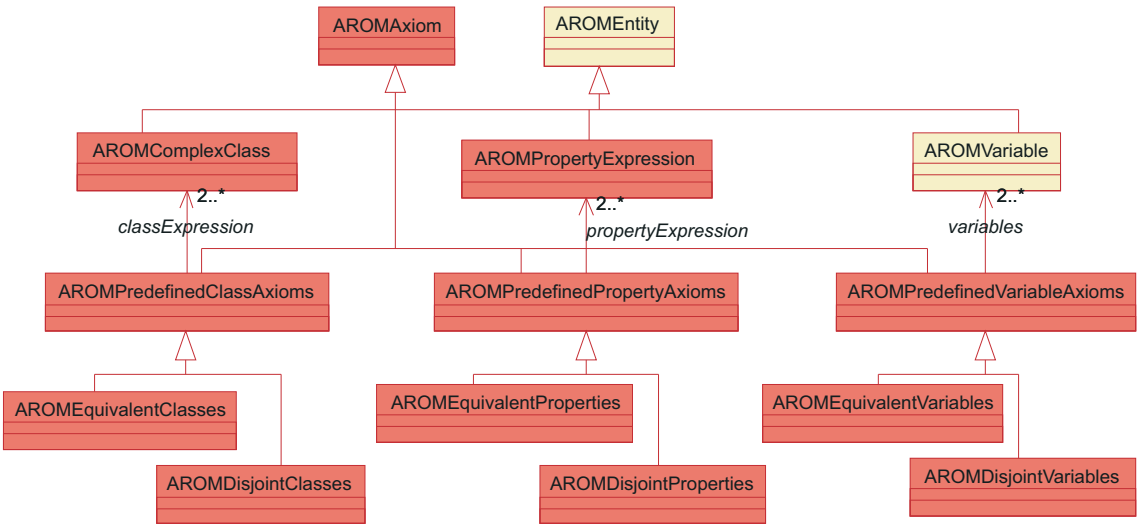


FIG. *5.33: Fragment du méta-modèle AROM-ONTO pour la définition des associations prédéfinies qui modélisent les axiomes d'équivalence et de disjonction.*

Au niveau du langage de description des bases de connaissances, nous avons ajouté de nouvelles primitives, `equivalent-classes` et `disjoint-classes` qui permettent la définition des axiomes d'équivalence et de disjonction (voir l'annexe B). La sémantique formelle de ces nouvelles constructions est définie dans l'annexe B.2.4.

Les axiomes d'équivalence et de disjonction entre les propriétés sont définis d'une façon similaire à celles des classes, la seule différence étant le fait qu'elles relient des expressions à base de propriétés et non pas des expressions à base de classes. Elles sont modélisées à travers la méta-classe `AROMPredefinedPropertyAxioms`, reliée par une relation générique de cardinalité minimum égale à deux, à la méta-classe des expressions à base de propriétés, `AROMPropertyExpression`. Cette relation est héritée par ses deux spécialisations, `AROMEquivalentProperties` et `AROMDisjointProperties`. Le langage de description de bases de connaissances a été étendu pour prendre en compte ces nouvelles structures de représentation, comme illustré dans l'annexe A. La fonction d'interprétation d'une base de connaissances a été également étendue pour prendre en compte la sémantique des nouvelles primitives (voir annexe B.2.4).

L'équivalence et la disjonction des variables est modélisée en AROM-ONTO à travers la méta-classe `AROMPredefinedVariableAxioms`, qui relie deux ou plusieurs variables, définies dans la fermeture transitive de la base de connaissances courante. Cette méta-classe est spécialisée par `AROMEquivalentVariables` et `AROMDisjointVariables` qui affinent ses spécifications pour qu'elles correspondent respectivement à la sémantique de la relation d'équivalence et de disjonction. La syntaxe du langage de description des bases de connaissances et sa sémantique formelle ont été étendues pour prendre en compte les nouvelles structures. La spécification de ces extensions peut être consultée dans les annexes A et B.2.4.

Comme dans le cas des axiomes d'égalité et de disjonction définis pour des objets, les axiomes d'équivalence et de disjonction introduits dans la section courante ne sont pas utilisés par le système AROM-ONTO pour vérifier la cohérence de la base de connaissances qui les introduit. En revanche, ces axiomes sont utiles pour garder la trace de déclarations correspondantes en OWL 2, ce qui garanti un minimum de perte d'information lors d'un processus d'importation/exportation d'une ontologie vers/depuis AROM-ONTO.

Propriétés inverses

Pour définir deux expressions à base de propriétés comme étant inverses, OWL 2 met a disposition un axiome dédié, `InverseObjectProperties`. Si par l'intermédiaire de l'expression `ObjectInverseOf`, on peut désigner de façon générique l'inverse d'une propriété OWL 2, l'axiome `InverseObjectProperties`, défini pour deux expressions à base de propriétés $\texttt{PE}_1$ et $\texttt{PE}_2$, établit une relation d'équivalence entre $\texttt{PE}_1$ et $\texttt{ObjectInverseOf}(\texttt{PE}_2)$ et entre $\texttt{ObjectInverseOf}(\texttt{PE}_1)$ et $\texttt{PE}_2$.

En AROM-ONTO, nous avons modélisé cet axiome de propriétés à travers la méta-classe `InverseOf`, dont les instances sont des paires d'expressions à base de propriétés. Les caractéristiques des propriétés inverses, ainsi que les méthodes de vérification de la cohérence de la base de connaissance et les méthodes d'inférence des conséquences logiques d'une telle déclaration sont implémentées dans le code de création des axiomes de type `InverseOf`. Concrètement, il s'agit de fonctionnalités pour vérifier que :

– chaque tuple de type `InverseOf` relie exactement deux expressions à base de propriétés AROM-ONTO, définies dans la fermeture transitive de la base de connaissances courante ;

– si les tuples $\langle PE_1, PE_2 \rangle$ et $\langle PE_1, PE_3 \rangle$ sont des instances de l'axiome `InverseOf`, alors on peut déduire que PE_2 et PE_3 sont équivalentes (au sens où leur extensions sont égales) ;
– si le tuple $\langle x, y \rangle$ appartient à l'extension de l'expression à base de propriétés PE_1, alors le tuple $\langle y, x \rangle$ appartient à l'extension de l'expression à base de propriétés PE_2, si $\langle PE_1, PE_2 \rangle$ sont reliés par une axiome de type `InverseOf`.

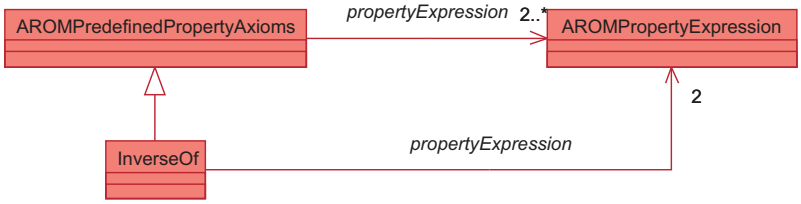


FIG. *5.34: Diagramme des classes UML qui définit la structure des expressions à base de propriétés en AROM.*

Nous avons vu les principales modifications qui doivent être opérées au niveau du méta-modèle de AROM afin de rendre son expressivité comparable à celle du langage OWL 2 DL. À présent, nous présentons la comparaison des deux langages du point de vue des systèmes de typage qu'ils offrent et du point de vue des inférences.

5.4.4 Typage

Les systèmes de typage utilisés par les deux langages sont très différents. AROM définit son propre module de types qui est puissant et facilement extensible. Au contraire, OWL 2 utilise le schéma XML [51] qui offre des types de base qui, même s'ils sont extensibles, ne peuvent pas être référencés depuis une ontologie. Le choix d'un IRI qui identifie le nouveau type est le problème principal rencontré lors de la description d'une ressource à l'aide d'un type défini par l'utilisateur à l'aide d'un schéma XML. De plus, les raisonnements à base de types étendus, tels que les types spatiaux ou temporels, ne sont pas toujours décidables (voir section 3.2.2.3), et la grande partie des raisonneurs ne prennent pas en compte les types définis.

Les types de base proposés par les deux langages sont analogues, mais l'avantage d'AROM est que, à partir des types simples, il permet l'introduction de restrictions ou de définitions de structures complexes avec un effort réduit de la part de l'utilisateur [192]. Les types d'AROM sont des classes Java qui implémentent une algèbre ensembliste en plus de leur interface propre. Un autre avantage d'AROM est qu'il permet la réutilisation des types issus des bibliothèques *open source* avec un minimum de modifications. En revanche, la grande expressivité de OWL 2 est réduite par un système de typage inflexible.

5.4.5 Inférences

Le standard OWL 2 permet la modélisation des ontologies, en imposant certaines règles syntaxiques et sémantiques. Des inférences peuvent être opérées à partir de faits et d'axiomes contenus dans les domaines modélisés en utilisant des raisonneurs externes, en majorité basés sur les *Logiques de Description.* La plupart des raisonnements concerne l'inférence de subsomption, ou la vérification de consistance de concepts. Par ailleurs, le traitement des restrictions sur les cardinalités des classes ou propriétés, le test d'égalité entre des chaînes de caractères, ou la résolution

des équations polynomiales sur des réels ou des cardinaux, sont des raisonnements algébriques acceptés.

En revanche, AROM offre les fondements pour la définition d'équations complexes qui décrivent les valeurs des variables à partir d'autres variables appartenant à la même classe ou à des classes différentes, via le Langage de Modélisation Algébrique (LMA) [176]. Ces relations de dépendance sont très puissantes et facilitent beaucoup le travail de l'utilisateur et la maintenance de la cohérence d'une base de connaissances. Étant extensible, le LMA donne la possibilité d'introduire des opérateurs adaptés aux besoins de l'utilisateur et aux types de données utilisés.

Dans les cas où l'expressivité du LMA ne suffit pas, AROM offre la possibilité d'utiliser des procédures externes, écrites en Java, donnant tout le pouvoir d'inférence nécessaire pour modéliser un domaine ontologique. Enfin, le processus de classification des instances (objets et tuples) d'AROM est un autre avantage de celui-ci. Les raisonneurs que nous avons étudiés (RacerPro et Pellet) n'offrent pas de raisonnement similaire.

5.5 Le traducteur AROM-ONTO/OWL 2 DL

Le moteur de traduction entre une ontologie OWL 2 DL et une base de connaissances AROM-ONTO est basé sur l'utilisation d'un ensemble de règles XSLT [59]. Ces règles exploitent la syntaxe RDF/XML utilisée par OWL 2, ainsi que la syntaxe XML de AROM.

5.5.1 Traduction OWL 2 DL → AROM-ONTO

La traduction d'une ontologie OWL 2 DL vers AROM-ONTO se fait en trois étapes :

- la première, qui est aussi la plus difficile, consiste à rassembler des descriptions des entités OWL 2 (classes, propriétés et individus) qui peuvent être distribuées à des endroits différents dans l'ontologie source, afin qu'elles puissent être regroupées dans les structures AROM-ONTO correspondantes.
- la seconde étape produit une base de connaissances AROM-ONTO dans laquelle chaque classe/association du modèle qui possède des super-classes/super-associations est précédée par la description de ses super-classes/super-associations.
- la dernière étape élimine les doublons introduits par l'étape précédente, en gardant exclusivement la première occurrence de chaque description.

Les instances OWL 2 qui appartiennent à plusieurs classes différentes, par exemple $(\texttt{Paul})^O \in (\texttt{Student})^C$ et $(\texttt{Paul})^O \in (\texttt{Scientist})^C$, sont traduites comme des instances d'une nouvelle classe `intersection-of(Student,Scientist)` (voir figure 5.35). Comme les expressions à base de classes de AROM-ONTO sont *abstraites* dans le sens où elles ne possèdent pas d'instances propres (voir section 5.4.3), nous introduisons une nouvelle classe - dans l'exemple choisi `extension1`- comme sous-classe de `intersection-of(Student,Scientist)` qui garde les instances communes à `Student` et `Scientist`.

5.5.2 Traduction AROM-ONTO → OWL 2 DL

La traduction d'une base de connaissances AROM-ONTO vers une ontologie OWL 2 DL avec une perte minimale d'information pose quelques défis. Dans un premier temps, afin de traduire les relations n-aires en OWL 2, nous avons décidé d'utiliser la réification d'une association par

une classe et un ensemble de *n* propriétés-lien. Pour identifier les associations n-aire comme telles nous ajoutons l'estampille "N-ary association" à chaque classe OWL 2 qui réifie une association (voir la classe `studiesAt` de la figure 5.36). Les variables qui caractérisent l'association n-aire sont transformées dans des attributs ayant comme domaine la classe qui représente l'association.

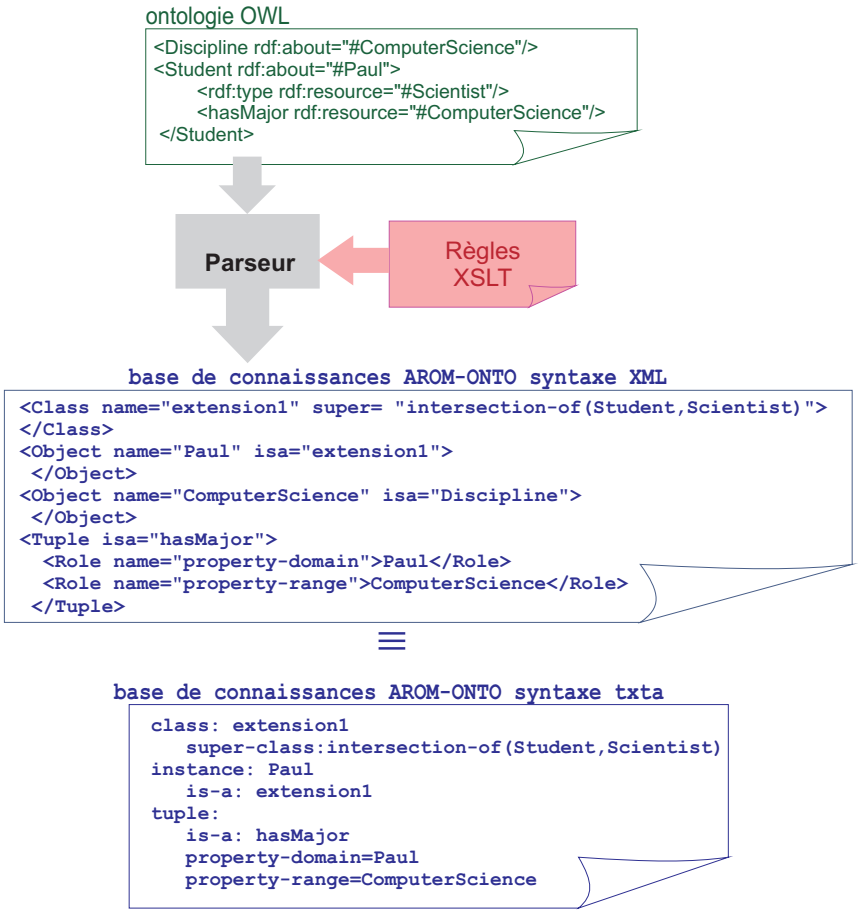


FIG. *5.35: Exemple de traduction de OWL 2 DL vers AROM-ONTO.*

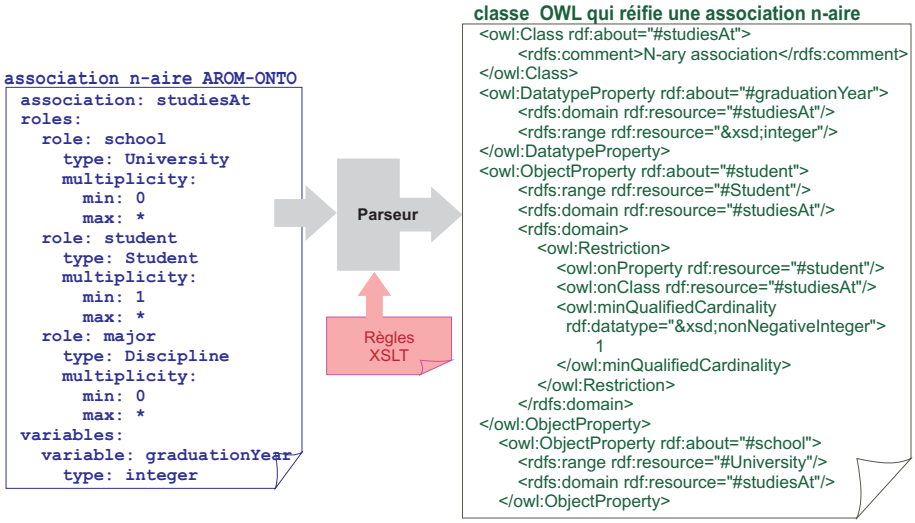


FIG. *5.36: Exemple de traduction d'une association n-aire de AROM-ONTO vers OWL 2.*

Pour traduire les restrictions de type, ainsi que les spécifications algébriques attachées aux variables AROM-ONTO, nous utilisons des commentaires OWL 2(`rdfs:comment`), comme illustré dans la figure 5.37.

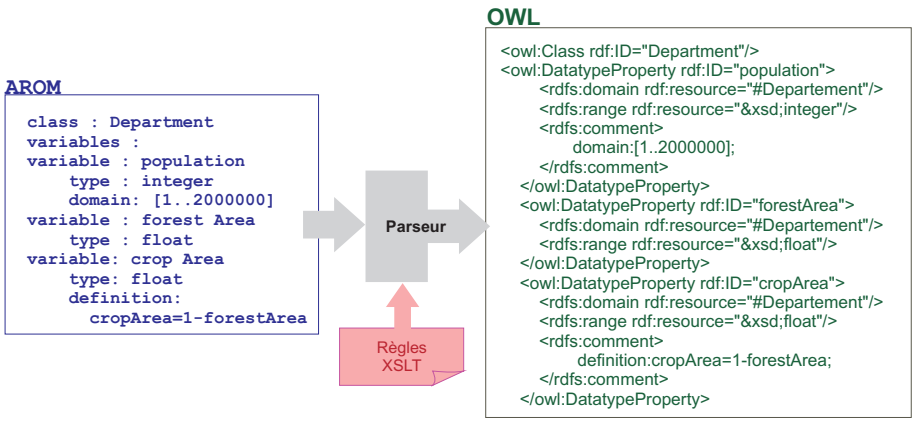


FIG. *5.37: Traduction d'une classe dont la définition contient des équations de AROM-ONTO vers OWL 2 DL.*

Dans la version actuelle, les deux *prototypes* de fichiers de règles XSLT que nous avons définis prennent en charge seulement un ensemble limité de structures de représentation (les classes simples, les classes complexes de type conjonction, disjonction, et négation, les variables et les propriétés). Néanmoins, comme perspective à court terme, nous entendons compléter l'ensemble de règles utilisées par les traducteurs AROM-ONTO → OWL 2 DL et OWL 2 DL → AROM-ONTO, pour prendre en compte toutes les structures de représentation offertes par les deux langages.

5.6 Synthèse

Dans ce chapitre nous avons présenté AROM-ONTO, une extension du Système de Représentation des Connaissances par Objets AROM. Cette extension est issue d'une comparaison entre AROM et OWL 2 DL et du constat de la complémentarité des deux langages, du point de vue des systèmes de typage et des raisonnements qu'ils offrent. Voulant profiter de cette complémentarité, nous avons défini un pont entre les deux langages, matérialisé par une extension du méta-modèle de AROM qui garantit leur compatibilité du point de vue de la représentation. Nous avons construit cette extension du système AROM, afin de garantir un minimum de perte d'informations lors de l'importation/l'exportation d'une ontologie dans/depuis AROM-ONTO. Ainsi, avec AROM-ONTO, on peut accéder dorénavant à des inférences proposées par des Logiques de Description (qui peuvent être réalisées en passant par des raisonneurs tels que Pellet, RacerPro *etc.*) et on peut également enrichir les ontologies OWL DL/OWL 2 DL à travers des raisonnements AROM, qui sont complémentaires par rapport à ceux proposés par les raisonneurs compatibles OWL 2.

Le nouveau système AROM-ONTO peut être utilisé comme raisonneur algébrique pour le Web Sémantique. Pour bénéficier des capacités d'inférence offertes par AROM-ONTO, les connaissances ontologiques doivent être traduites vers le formalisme propre à AROM. À cet effet, nous avons construit un traducteur à base de règles XSLT qui permet de importer des ontologies OWL 2 DL en AROM-ONTO, mais aussi d'exporter des bases de connaissances AROM-ONTO en OWL 2 DL.

Cette proposition est une solution générique, qui ne dépend pas d'un domaine d'application spécifique, pour étendre l'ensemble des raisonnements disponibles pour le Web Sémantique. Elle représente le fondement sur lequel nous avons construit un raisonneur spatial et temporel capable d'exploiter des ontologies OWL 2 DL, et présenté dans le chapitre suivant.

Chapitre 6

Le système ONTOAST

People mistakenly assume that their thinking is done by their head ; it is actually done by the heart which first dictates the conclusion, then commands the head to provide the reasoning that will defend it.

Anthony de Mello

SOMMAIRE

Résumé

Ce chapitre présente ONTOAST, un système de représentation et de raisonnement spatial et temporel quantitatif et qualitatif, construit comme extension du système AROM-ONTO. ONTOAST peut être utilisé dans le contexte du Web Sémantique Géospatial pour exploiter les annotations spatiales et temporelles décrites en OWL DL ou OWL 2 DL en utilisant les types temporels proposés par les deux langages et/ou les concepts spatiaux et temporels définis par les ontologies GeoRSS-Simple, OWL-Time et QualitativeSpatialRelations.

6.1 Introduction

Dans ce chapitre, nous proposons la construction, au dessus du système AROM-ONTO, d'un module de représentation et de raisonnement spatial et temporel, appelé ONTOAST, capable de décrire et d'exploiter des informations spatiales et temporelles à la fois quantitatives et qualitatives. ONTOAST est donc une extension spatio-temporelle du système AROM-ONTO défini dans le chapitre 5. L'un des avantages principaux de ONTOAST, par rapport à la problématique identifiée dans le chapitre 3, vient de son ouverture vers le Web Sémantique Géospatial garantie par AROM-ONTO (voir section 6.2). Ainsi, le raisonneur que nous proposons ici peut être utilisé pour exploiter les informations spatiales et temporelles quantitatives et qualitatives définies dans des ontologies OWL DL ou OWL 2DL, suite à leur traduction préalable dans ONTOAST.

À partir du moment où les connaissances ontologiques sont importées dans ONTOAST, toute la palette de raisonnements dont il dispose peut être utilisée pour compléter l'ontologie avec les connaissances implicites. Ensuite, par le biais de l'exportation des résultats de calculs dans OWL, ces derniers peuvent être exploités dans le cadre général du Web Sémantique.

Nous proposons également la définition d'un ensemble de relations qualitatives géospatiales pour ONTOAST, afin d'accroître sa flexibilité et son expressivité, ainsi que pour permettre des raisonnements lorsqu'on ne dispose pas de données spatiales et temporelles précises (voir sections 6.4 et 6.5). Des associations géospatiales complexes pourront désormais être définies à l'aide d'équations exprimées dans le Langage de Modélisation Algébrique d'AROM, étendu ici par des opérateurs de topologie, d'orientation et de proximité des objets (voir section 6.3). Bénéficiant de la compatibilité entre AROM-ONTO et OWL 2 DL, ONTOAST permet donc de décrire des ontologies géospatiales pour le Web Sémantique Géospatial, ainsi que de formuler et de traiter des requêtes qualitatives sur ces ontologies, en exploitant les mécanismes de raisonnement d'AROM.

La construction du Web Sémantique Géospatial est soutenue par la mise à disposition, à travers l'utilisation du système ONTOAST, d'un ensemble de raisonnements spatiaux et temporels qui peuvent exploiter les connaissances décrites en OWL DL/OWL 2 DL, notamment :

- la déduction de relations spatiales/temporelles qualitatives à partir de données quantitatives connues ; Ce type d'inférence est basé sur les définitions des relations spatiales et temporelles qualitatives (voir sections 2.3.2, 2.4.2) et utilise des calculs mathématiques pour traduire les données numériques en relations qualitatives ;
- la déduction de relations spatiales/temporelles qualitatives à partir d'autres relations spatiales/temporelles explicitement stockées dans la base ontologique ou déduites (voir sections 6.4.4 et 6.5.4). Ces raisonnements sont construits sur les propriétés des relations spatiales/temporelles, telles que la transitivité, la symétrie, *etc.*, ainsi que sur les tableaux de composition spécifiques à chaque type de relation qualitative (voir chapitre 2) ;
- les raisonnements algébriques à base d'équations.

6.2 De AROM-ONTO vers ONTOAST

Le système ONTOAST est construit comme extension spatiale et temporelle du système AROM-ONTO. Si AROM-ONTO garantit la compatibilité générale entre AROM et les langages d'ontologies OWL DL et OWL 2 DL, ONTOAST propose l'extension des traducteurs OWL↔AROM-ONTO pour prendre en compte de façon adaptée les descriptions spatiales et temporelles réalisées par rapport aux ontologies *GeoRSS-Simple* et *OWL Time*. Celle-ci est mise en œuvre à travers la

définition d'un ensemble de règles de traduction qui prennent en compte la spécificité des informations spatiales et temporelles que ONTOAST propose.

Ainsi, les raisonnements spatiaux et temporels que nous définissons dans ce chapitre, peuvent être utilisés pour exploiter les descriptions spatiales et temporelles définies dans des ontologies OWL DL et OWL 2 DL, moyennant une traduction vers et depuis ONTOAST. Un aperçu de cette chaîne de traitements est présenté dans la figure 6.1.

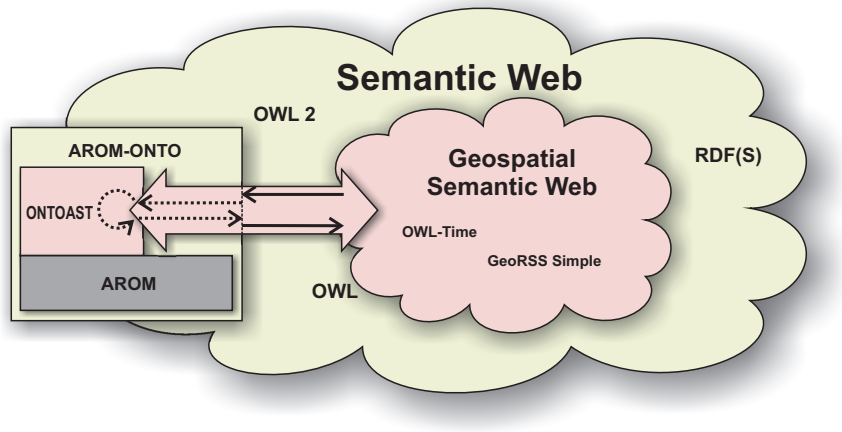


FIG. *6.1: Modifications apportées par ONTOAST à l'architecture initiale de AROM.*

ONTOAST apporte plusieurs modifications au système AROM-ONTO, qui sont localisées au niveau des modules d'*Implémentation du Modèle Objet AROM de Types*, et d'*Interprétation Algébrique* (voir figure 5.3, page 116). Au niveau du *Module de Types*, et du *Module d'Interprétation Algébrique*, nous avons complété l'extension AROM-ST (décrite dans la section 6.3.1) par un ensemble d'opérateurs spatiaux et temporels qui facilitent la transformation des données quantitatives dans des relations qualitatives. Nous proposons également un modèle spatial et temporel prédéfini qui est présenté dans la section 6.4.

6.3 Modèle spatial et temporel quantitatif de ONTOAST

6.3.1 Extension AROM-ST [174]

Le module de types d'AROM définit l'ensemble des types de données et d'opérations sur ces types qui peuvent être utilisés dans une base de connaissances. De la même manière que le système de types Metéo [48], le module de types d'AROM considère deux niveaux de représentation des types :

- le $C-type$ (classe de types) qui représente l'ensemble, fini ou non, des valeurs partageant une même structure, tel que l'ensemble des réels ou l'ensemble des chaînes de caractères. Chaque $C-type$ définit des opérations qui lui sont applicables ;
- les $\delta-types$ qui représentent des sous-ensembles de $C-types$. Chaque $\delta-type$ est associé à un $C-type$ qui représente son type principal. Les informations relatives aux données et les opérations se trouvent donc dans le $C-type$. En revanche, les $\delta-types$ contiennent des informations relatives à la restriction du domaine défini par le $C-type$. Pour un même $C-type$ il peut donc exister plusieurs, voir une infinité, de $\delta-types$.

En exploitant l'extensibilité du *Module de Types* du système AROM, Moisuc a proposé dans sa thèse [174] l'extension AROM-ST qui intègre le modèle spatial vectoriel proposé par l'OGC (décrit dans la section 2.3.1.1). Selon cette norme, l'ensemble des types spatiaux nécessaires et suffisants pour représenter tout objet spatial, est composé des types géométriques simples : `point`, `polyline` et `polygon` (voir figure 6.2). Les types `line` et `linearRing` sont définis en appliquant des contraintes sur le type `polyline`. Comme illustré dans la figure 6.2, à partir des types simples sont définis les types géométriques complexes (composés) : `multiPoint` (nuage de points), `multiLine` et `multiArea`.

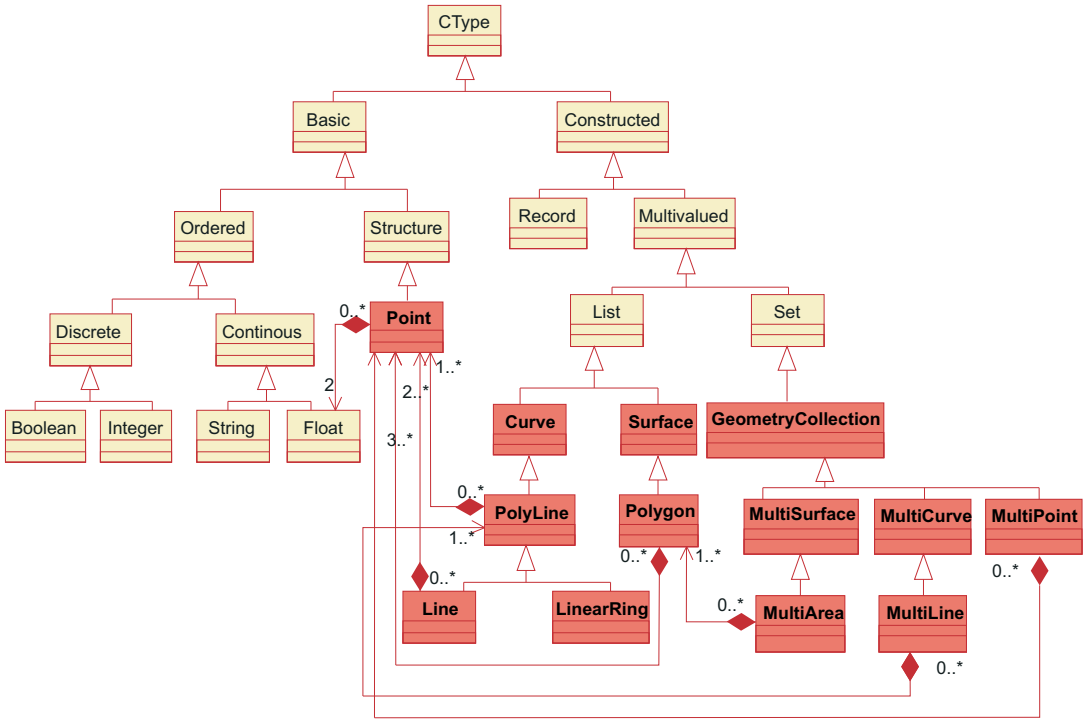


FIG. *6.2: Méta-modèle définissant l'ensemble de types géométriques proposé par AROM-ST.*

Les types temporels simples en AROM-ST sont le type `instant` et le type `interval` (voir figure 6.3). Les types composés, constituant des collections d'*instants* ou d'*intervalles*, sont les types `multiInstant` et `multiInterval`.

À cet ensemble de types défini par [174], nous avons ajouté le type `duration`, afin de permettre la modélisation des durées. Le type `duration` de AROM-ST est construit comme une adaptation du type homonyme défini par la librairie `javax.xml.datatype.Duration`.

Afin de formuler des requêtes, d'exprimer des contraintes, ou bien de décrire des inférences sur des variables de types spatio-temporels, le LMA d'AROM a été également étendu [174], en introduisant trois catégories d'opérateurs spatiaux et temporaux :

a) les *opérateurs topologiques* qui sont des prédicats logiques binaires testant la position relative (spatiale ou temporelle) de deux objets. Les opérateurs topologiques spatiaux introduits sont ceux définis par le consortium OpenGIS : `contains`, `within`, `intersects`, `disjoint`, `overlaps`, `crosses` et `equals`. Pour compléter la totalité des positions spatiales possibles, l'opérateur `inAdjacent` a été ajouté [85]. Ces opérateurs sont définis pour des polygones (à l'exception de `crosses`, défini pour un objet de type `line` qui peut traverser une région sur-

facique) mais peuvent s'appliquer à des paramètres moins complexes. En ce qui concerne la topologie temporelle, l'ensemble d'opérateurs définis par AROM-ST est l'image fidèle des relations proposées par Allen [14] : `before`, `after`, `starts`, `started-by`, `finishes`, `finishedby`, `during`, `contains`, `equals`, `meets`, `met-by`, `overlaps`, `overlapped-by` ;

b) les *opérateurs ensemblistes* qui traitent l'*espace* et le *temps*, respectivement comme un ensemble de *points* et d'*instants* temporels. Ces opérateurs ont la même forme pour la dimension spatiale et pour la dimension temporelle : `union`, `intersection`, `difference` et `symmetricalDifference` ;

c) les *opérateurs de mesure* qui donnent une description quantitative de l'*espace* et du *temps*. Pour la dimension spatiale, les opérateurs définis sont `dimension`, pour mesurer la surface ou l'étendu d'une région, et `distance`, pour mesurer l'écartement ou l'éloignement entre deux régions, leurs correspondants pour la dimension temporelle étant `duration` et `timeLength`.

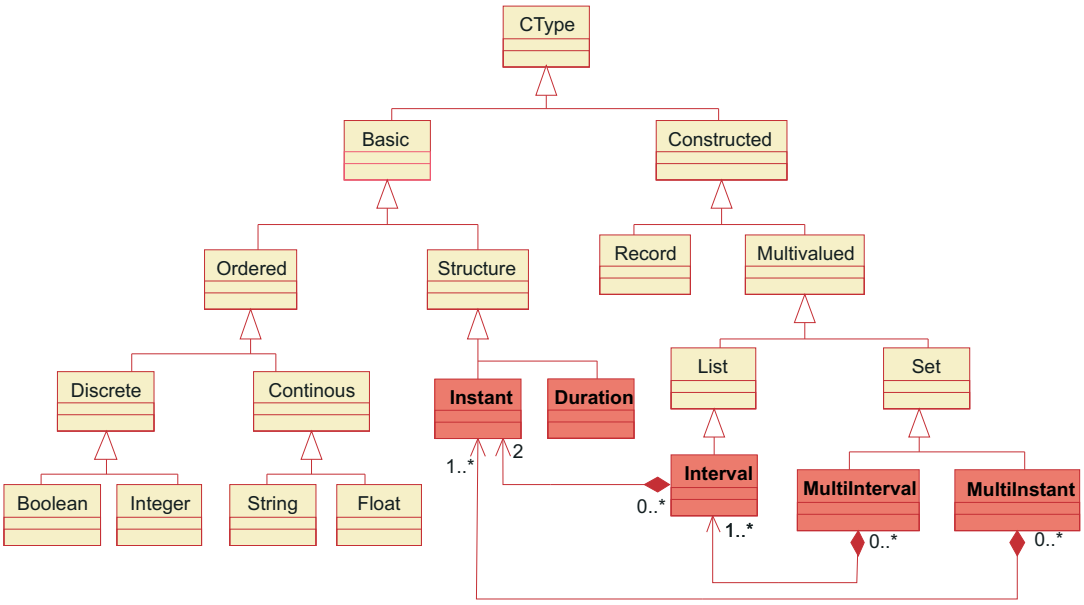


FIG. *6.3: Méta-modèle définissant l'ensemble des types temporels proposés par AROM-ST.*

Nous avons complété cet ensemble d'opérateurs, en ajoutant :

– l'*opérateur de direction*, `direction`, qui teste l'orientation relative d'un objet spatial *A* par rapport à un objet spatial *B*. Les objets *A* et *B* peuvent être représentés par des polygones ou par des structures moins complexes (lignes ou points). L'opérateur `direction` renvoie une relation de direction de type *carreau unique* (définie dans la section 2.3.2.4), parmi `North`, `West`, `South`, `East` et `Neutral`, si la géométrie de l'objet *A* est incluse dans un seul carreau de direction construit à partir du rectangle englobant minimal de l'objet de référence *B*. Dans le cas contraire, on obtient une relation de direction de type *carreaux multiples*, telle que `North:West:Neutral` ou `South:East`, *etc.*

– les *opérateurs de distance* `veryClose`, `close`, `far` et `veryFar` sont définis par rapport à des échelles.

6.3.2 Ontologie de types spatiaux utilisée par ONTOAST

Afin de traduire les données spatiales décrites à l'aide de types spatiaux définis par AROM-ST (voir section 6.3.1) vers OWL/OWL 2, nous utilisons *GeoRSS-Simple* (présentée dans la section 3.3.1) comme ontologie de référence. Dans les sous-sections qui suivent, nous définissons les correspondances entre les types de données de AROM-ST et les concepts spatiaux et les attributs introduits par GeoRSS-Simple.

6.3.2.1 AROM-ST → GeoRSS-Simple

Les types spatiaux introduits par AROM-ST sont utilisés dans la définition de l'*intension* d'une classe ou d'une association ONTOAST. Ils n'ont pas de correspondants directs dans le langage OWL. Néanmoins, comme présenté dans la section 3.3.1, afin de décrire en OWL des données spatiales quantitatives, on peut utiliser des concepts et/ou attributs dont la sémantique est établie dans une ontologie considérée de référence.

Type AROM-ST	Concept GeoRSS-Simple	Attribut *GeoRSS-Simple*
point	–	`point`
line	–	`line`
polygon	–	`polygon`
polyLine	`gml:LineString`	–
linearRing	`gml:LinearRing`	–
multiPoint	–	`gml:posList`
multiLine	–	plusieurs attributs `line` et `qml:featuretypetag`="*MultiLine*"
multiArea	–	plusieurs attributs `polygon` et `qml:featuretypetag`="*MultiArea*"

TAB. *6.1: Correspondances entre les types AROM-ST et les concepts/attributs GeoRSS-Simple.*

Afin de garantir un minimum de perte d'information lors de la traduction d'une base de connaissances ONTOAST en une ontologie OWL 2 DL (ou OWL DL), nous définissons dans cette section les correspondances qui existent entre les types spatiaux proposés par AROM-ST et les structures proposées par l'ontologie *GeoRSS-Simple* que nous prenons comme référence. Le tableau 6.1 montre les différentes façons de traduire les données spatiales quantitatives pour le Web Sémantique. Par exemple, les valeurs de type `point`, `line` et `polygon` sont traduites en utilisant les attributs de type `xsd:string`, `point`, `line`, et respectivement `polygon`, définies dans l'ontologie *GeoRSS-Simple*.

La figure 6.4 présente l'exemple simple de la traduction de la classe `City` de ONTOAST vers OWL 2 DL (ou OWL DL). Cette classe ayant une variable dont le type est `polygon`, est traduite en OWL comme sous-classe de `gml:_Feature`. Elle hérite ainsi de la propriété `polygon`, qui ne doit pas être explicitement redéfinie, mais peut être directement utilisée pour décrire les instances de `City`. L'exemple de la figure 6.4 continue avec la traduction de la description attachée à la ville de `Cannes`. Il faut notamment observer la traduction de la variable `geom` et de sa valeur. Cette dernière n'a pas un attribut homonyme, mais correspond à l'attribut `polygon`. L'attribut utilisé pour traduire la valeur de la variable `geom` est choisi en fonction du type de la variable. Par exemple, si le type

de `geom` était `point`, sa valeur en OWL 2/OWL aurait été décrite à l'aide de l'attribut GeoRSS Simple `point`.

Dans ce travail, nous avons choisi de limiter à un le nombre de variables de type spatial qui peuvent être attachées à une classe/association. Néanmoins, cette limitation est assez restrictive et il serait intéressant d'étudier les possibilités de traduction de plusieurs variables spatiales qui modélisent, par exemple, la géométrie d'un objet selon plusieurs niveaux de détails.

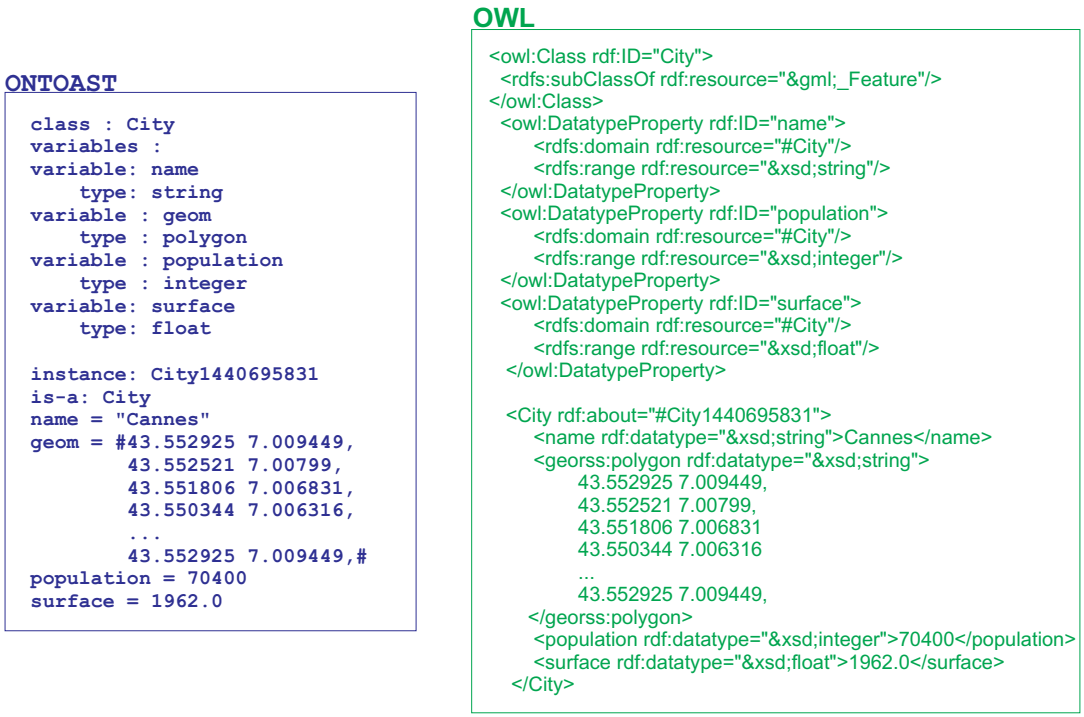


FIG. *6.4: Exemple de traduction de données spatiales de ONTOAST vers OWL-DL.*

Les valeurs de type `polyLine` et `linearRing` sont traduites en des instances de concepts `gml:LineString`, respectivement `gml:LinearRing`, reliées aux objets qu'elles décrivent à en utilisant des instances de la propriété-lien `where`. Un ensemble de points (`multiPoint`) est traduit à l'aide d'une propriété `gml:posList`. Une valeur de type `multiLine` devient, lors de la traduction en OWL-DL, un ensemble de propriétés `line`. Pour signaler le fait que les valeurs des propriétés `line` attachées à un objet doivent être considérées comme un ensemble et non pas de façon individuelle, nous attribuons la valeur "*multiLine*" à la propriété `gml:featuretypetag`. Les valeurs de type `multiArea` sont modélisées de façon analogue.

6.3.2.2 GeoRSS-Simple → AROM-ST

En faisant le chemin inverse, si on souhaite accéder à l'ensemble des raisonnements spatiaux et temporels proposés par ONTOAST (que nous décrivons dans les sections 6.4 et 6.5) pour réaliser des inférences sur une ontologie OWL DL/OWL 2 DL, les données spatiales quantitatives que celle-ci décrit doivent être transformées en des valeurs concrètes attachées aux types spatiaux proposés par AROM-ST (voir section 6.3.1). Dans ce travail, nous considérons exclusivement les descriptions réalisées par rapport à l'ontologie GeoRSS-Simple. Néanmoins, il serait intéressant d'étudier des descriptions créées à l'aide d'autres ontologies proposant des concepts spatiaux et

dont les définitions sont alignées avec l'ontologie GeoRSS-Simple. Ce type d'approche pose un certain nombre de problèmes surtout concernant l'hétérogénéité des structures utilisées pour décrire les types spatiaux. Si un appariement du type :

```
<owl:Class rdf:about="&gml;Point">
        <owl:equivalentClass rdf:resource="#exPoint"/>
</owl:Class>
```

garantit le fait que les extensions des deux classes (`Point` et `exPoint`) sont équivalentes, il ne spécifie pas les règles de transformation des données d'un format vers l'autre.

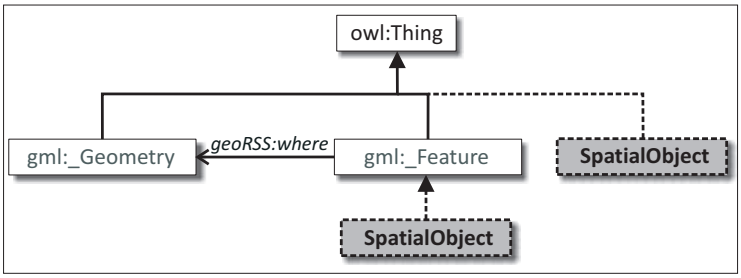


FIG. *6.5: La position qu'un concept spatial (`SpatialObject`) peut avoir par rapport aux classes gml :_Feature et gml :_Geometry définies par l'ontologie GeoRSS-Simple.*

Comme le montre la figure 6.5, pour définir un concept spatial (`SpatialObject`) en OWL 2/OWL à l'aide de l'ontologie GeoRSS-Simple, on a plusieurs alternatives :

- on peut lui attacher exclusivement une description thématique et profiter des attributs qu'il hérite de `owl:Thing` – `gml:upperCorner`, `gml:lowerCorner`, `gml:pos` et `gml:posList` (voir la définition de l'ontologie GeoRSS-Simple dans la section 3.3.1)– pour situer ses instances dans l'espace ;
- on peut le définir comme sous-classe de `gml:_Feature` et décrire les caractéristiques spatiales de ses instances en utilisant les attributs définis pour `gml:_Feature` ;
- on peut le définir comme sous-classe de `gml:_Feature` et utiliser la relation `where` pour lui associer une instance de la classe `gml:_Geometry`.

Pour les trois alternatives énumérées auparavant, le résultat de la traduction en ONTOAST est le même, à ceci près que le type de la variable `geom` change en fonction du type de géométrie attaché aux instances de la classe `SpatialObject`. Il faut préciser qu'en OWL rien n'empêche l'utilisateur de définir pour les instances d'une même classe des géométries hétérogènes. Par exemple, une instance x de `SpatialObject` peut être décrite par un attribut `polygon` alors qu'une autre instance y de la même classe peut être décrite par une propriété-lien `where` avec une instance de la classe `gml:Point`. Dans ce cas, la traduction en ONTOAST de la classe `SpatialObject` doit contenir deux variables spatiales : l'une de type `polygon` et l'autre de type `point`. Néanmoins, pour des raisons de clarté de la présentation, dans cette thèse, nous faisons la supposition que ce genre de situation ne peut pas arriver.

Supposons qu'on veut traduire vers OWL une classe spatiale A' qui a été traduite en ONTOAST au préalable à partir d'une classe OWL A. Pour garantir le fait que le résultat de la traduction, A'', est identique à la définition initiale A, on garde une trace de l'origine de la variable spatiale `geom`. En d'autre mots, en ONTOAST, à chaque variable spatiale obtenue suite à une traduction depuis

OWL, correspond l'un des tags : "where", "feature", "thing", définis à l'aide de la facette de documentation. L'exemple de la figure 6.6 illustre la définition d'un tel tag. Ces tags sont utilisées lors de la traduction vers OWL d'une classe ONTOAST disposant d'une variable spatiale. Concrètement, si la classe A' dispose d'une variable `geom` de type `polygon`, dont le tag est "where", les caractéristiques spatiales de ses instances sont traduites en OWL comme des instances de la classe `gml:_Geometry` reliés par des propriétés `where` aux instances de A'. Si la variable geom n'a pas de tag, alors elle est traduite selon les règles définies dans le tableau 6.1.

OWL

```
<owl:DatatypeProperty rdf:about="qsr#name">
      <rdfs:domain rdf:resource="qsr#City"/>
      <rdfs:range rdf:resource="&xsd;string"/>
</owl:DatatypeProperty>

<City rdf:about="#city1">
      <name>Cannes</name>
      <georss:where rdf:resource="#polygon1"/>
</City>

<gml:Polygon rdf:about="#polygon1">
      <gml:exterior rdf:resource="#linearRing1"/>
</gml:Polygon>

<gml:LinearRing rdf:about="#linearRing1">
      <gml:posList rdf:datatype="&xsd;string">
            43.552925 7.009449,
            43.552521 7.00799,
            43.551806 7.006831
            43.550344 7.006316
            ...
            43.552925 7.009449,
       </gml:posList>
</gml:LinearRing>
```

ONTOAST

```
class City
   variables:
     variable: name
       type: string
     variable: geom
       documentation: "where"
       type: polygon

instance: city1
is-a: City
name = "Cannes"
geom = #43.552925 7.009449,
        43.552521 7.00799,
        43.551806 7.006831,
        43.550344 7.006316,
        ...
        43.552925 7.009449,#
```

FIG. *6.6: Exemple de traduction de données spatiales quantitatives définies par rapport à l'ontologie GeoRSS-Simple vers ONTOAST.*

Concept GeoRSS-Simple	Type AROM-ST
`gml:Point`	point
`gml:LineString`	line
`gml:Polygon`	polygon
`qml:LinearRing`	linearRing
`gml:Envelope`	polygon

TAB. *6.2: Correspondances entre les concepts spatiaux utilisés par GeoRSS-Simple et les types AROM-ST.*

Pour traduire les structures spatiales proposées par GeoRSS-Simple vers ONTOAST, nous avons défini des règles de transformation. Ces règles sont synthétisées dans les tableaux 6.2 et 6.3. Les instances des classes `gml:Point`, `gml:Line`, `gml:Polygon` et `gml:LinearRing` sont transformées en des valeurs concrètes de types `point`, `line`, `polygon` et `linearRing` respectivement. Une telle valeur concrète est ajoutée à la définition de l'objet spatial relié par une propriété `where` à la description considérée. Pour illustrer notre démarche, nous présentons dans la figure 6.6 un

exemple de traduction. La description du contour géographique de la ville de Cannes est définie comme instance (`linearRing1`) de la classe `gml:LinearRing`. Ce contour précise l'extérieur du polygone `polygon1`, qui définit les limites géographiques de la ville de Cannes. Le lien entre l'instance `city1` et `polygon1` est réalisé à travers la relation `where`.

La définition ONTOAST équivalente (voir figure 6.6) est beaucoup plus simple. L'extérieur du polygone `polygon1` devient une valeur concrète pour la variable `geom` qui caractérise l'objet `city1`. Comme on peut l'observer sur la figure 6.6, un polygone est spécifié en ONTOAST directement à travers l'ensemble de points qui définit sa courbe, alors que la description `gml` est plus complexe.

Attribut GeoRSS-Simple	Type AROM-ST
`point`	point
`line`	line
`polygon`	polygon
`box`	polygon
`lat` et `long`	point
`elev`	–
`floor`	–
`radius`	float
`gml:pos`	point
`gml:posList`	multiPoint
`gml:lowerCorner` et `gml:upperCorner`	polygon

TAB. *6.3: Correspondances entre les attributs définis par GeoRSS-Simple et les types AROM-ST.*

OWL

```
<owl:DatatypeProperty rdf:about="qsr#name">
      <rdfs:domain rdf:resource="qsr#City"/>
      <rdfs:range rdf:resource="&xsd;string"/>
</owl:DatatypeProperty>

<City rdf:about="#city1">
      <name>Cannes</name>
      <georss:where rdf:resource="#envelope1"/>
</City>

<gml:Envelope rdf:about="#envelope1">
      <gml:lowerCorner rdf:datatype="&xsd;string">
           43.540452  6.984558
      </gml:lowerCorner>
      <gml:upperCorner rdf:datatype="&xsd;string">
           43.553392 7.032623
      </gml:upperCorner>
</gml:Envelope>
```

ONTOAST

```
class City
   variables:
     variable: name
       type: string
     variable: geom
       documentation: "where"
       type: polygon

instance: city1
is-a: City
name = "Cannes"
geom = #43.540452 6.984558,
        43.540452 7.032623,
        43.553392 7.032623,
        43.553392 6.984558,
        43.540452 6.984558,#
```

FIG. *6.7: Exemple de traduction d'un rectangle englobant minimal de OWL vers ONTOAST.*

Une instance de la classe `gml:Envelope` est traduite en ONTOAST comme une donnée de type `polygon`. À partir des coordonnées attachées à un rectangle englobant minimal (instance de `gml:Envelope`), à travers les attributs `gml:lowerCorner` et `gml:upperCorner`, on peut construire un polygone (rectangle) comme illustré dans la figure 6.7.

Les correspondances entre les attributs définis par GeoRSS-Simple et les types spatiaux de ONTOAST sont présentées dans le tableau 6.3. Plusieurs illustrations de ces correspondances directes ont été décrites dans la section 6.3.2.1. Les attributs `elev` et `floor` ne peuvent pas être traduites de façon immédiate vers ONTOAST, car les types spatiaux de ATOM-ST ne prennent pas en compte l'élévation. Néanmoins, les attributs peuvent être traduits par des variables de même nom de type `float`, respectivement `integer`.

6.3.3 Ontologie de types temporels utilisée par ONTOAST

Afin de traduire dans OWL les données temporelles décrites à l'aide de types temporels définis par AROM-ST (voir section 6.3.1), nous utilisons les types temporels proposés par OWL mais aussi les structures temporelles définies par l'ontologie OWL-Time (présentée dans la section 3.4.2).

6.3.3.1 AROM-ST → OWL + OWL-Time

La traduction vers OWL de données temporelles décrites à l'aide de types temporels de ONTOAST, est résumée dans le tableau 6.4. Le type `instant` proposé par AROM-ST a comme correspondant le type `xsd:dateTime` de OWL. Les valeurs de ces types sont ainsi faciles à traduire entre les deux formalismes. En revanche, on ne dispose pas en OWL, de types dédiés permettant la manipulation explicite des intervalles. La traduction depuis ONTOAST se fait alors à l'aide de concepts et propriétés proposés par l'ontologie OWL-Time (voir section 3.4.2), devenue recommandation du W3C pour la description du temps pour le Web Sémantique .

Type AROM-ST	Type OWL	Classe/propriété OWL-Time
instant	`xsd:dateTime`	–
interval	–	`time:Interval` + `time:hasBeginning` + `time:hasEnd` `time:Instant`+ `time:inXSDDateTime`
multiInterval	–	–
multiInstant	–	–
duration	–	`time:DurationDescription`

TAB. *6.4: Correspondances entre les types temporels de AROM-ST et les concepts/propriétés définis par OWL-Time.*

Concrètement, pour traduire un intervalle de temps, modélisé en ONTOAST comme une paire d'instants, on utilise une instance de la classe `time:Interval` à laquelle on attache deux instances de la classe `time:Instant` en utilisant les propriétés-lien `time:hasBeginning` et `time:hasEnd` (voir l'exemple de la figure 6.8). Ensuite, chacun des deux instants sera décrit par un attribut `time:inXSDDateTime`.

Les types `multiInstant` et `multiInterval` ne trouvent pas d'équivalents dans l'ensemble de types proposé par OWL, ni dans l'ensemble de structures de représentation proposé par OWL-Time. Néanmoins, ils peuvent être traduits comme des ensembles d'instants, respectivement d'intervalles, attachés aux objets qu'ils caractérisent à travers des propriétés `holds` (voir figure 3.9, page 80). Les durées de temps (valeurs de type `duration`) sont traduites comme des instances

de la classe `time:DurationDescription`, décrites par des attributs `time:years`, `time:weeks`, `time:months`, `time:days`, `time:hours`, `time:minutes` et `time:seconds`.

ONTOAST

```
class : CulturalEvent
variables :
variable: name
      type:string
variable: timetable
      type:interval

instance: concert1
is-a: CulturalEvent
name="Sticky & Sweet Tour"
timetable="19/07/2009,21:30:00-
           19/07/2009,23:30:00"
```

OWL

```
<CulturalEvent rdf:about="#concert1">
      <name rdf:datatype="&xsd;string">
          Sticky & Sweet Tour
      </name>
      <time:holds rdf:resource="#interval1"/>
</CulturalEvent>
<time:Interval rdf:about="#interval1">
      <time:hasBeginning rdf:resource="#interval1Beginning"/>
      <time:hasEnd rdf:resource="#interval1End"/>
</time:Interval>
<time:Instant rdf:about="#interval1Beginning">
      <time:inXSDDateTime rdf:datatype="&xsd;dateTime>
          2009-07-19T21:30:00
      </time:inXSDDateTime>
</time:Instant>
<time:Instant rdf:about="#interval1End">
      <time:inXSDDateTime rdf:datatype="&xsd;dateTime">
          2009-07-19T23:30:00
      </time:inXSDDateTime>
 </time:Instant>
```

FIG. *6.8: Exemple de traduction d'un intervalle de temps depuis ONTOAST vers OWL.*

6.3.3.2 OWL+OWL-Time → AROM-ST

Les types temporels de AROM-ST sont construits à partir du type `Date` défini dans la librairie `java.util`. Les données de type `Date` représentent des instants précis, décrits par l'année, le mois, le jour ainsi que l'heure, la minute et la seconde. Pour les dates dont la spécification n'est pas complète, le système remplit les champs absents par des valeurs par défaut. Donc, le type `Date` de java, et, en conséquence, le type `instant` de AROM-ST ne permettent pas de décrire des intervalles implicites ou répétitifs comme ceux spécifiés à l'aide des types `xsd:gYearMonth`, `xsd:gDay`, `xsd:gMonth`, *etc.* du XML Schéma. Pour supprimer cet inconvénient, on pourrait modifier la définition du type `instant`. Cependant, en tenant compte du fait que les types temporels répétitifs sont abandonnés dans OWL 2, nous avons choisi de ne pas opérer cette modification.

Des intervalles de temps peuvent être décrits en ONTOAST à l'aide du type `interval`, construit à partir de deux instants : l'instant de début et l'instant de fin (valeurs de type `instant`). Cependant, le type `interval` ne peut pas être utilisé pour modéliser des intervalles implicites, comme ceux décrits par les types `xsd:gDay`, `xsd:gMonth`, *etc.* de OWL. Par exemple, en ONTOAST, on peut décrire le mois de *Septembre* de l'année 2009, en spécifiant l'instant auquel il commence et celui auquel il finit (*i.e.* `1/09/2009,00:00:00-30/09/2009,23:59:59`), mais pas le mois de *Septembre* en général.

La répétition peut être simulée à l'aide du type `multiInterval`, mais ce dernier peut seulement contenir un nombre limité d'intervalles. En conclusion, il n'est pas possible de spécifier en ONTAOST un intervalle qui se répète tous les ans, tous les mois, ou tous les jours, etc. Le tableau 6.5 présente les correspondances qui existent entre les types temporels proposés par XML Schéma et repris par OWL et les types temporels définis par AROM-ST.

Type OWL	Type AROM-ST
xsd:dateTime	instant
xsd:date	instant
xsd:gYearMonth	-
xsd:gMonthDay	-
xsd:gDay	-
xsd:gMonth	-

TAB. *6.5: Correspondances entre les types temporels de OWL et ceux de ONTOAST.*

Le tableau 6.6 illustre les différentes descriptions temporelles, réalisées par rapport à l'ontologie OWL-Time, qui peuvent être traduites comme des intervalles et/ou des instants AROM-ST. Les structures annotées par * doivent fournir des descriptions temporelles complètes, afin que la conformité de leur traduction puisse être garantie. Cela veut dire que, dans le cas d'un intervalle défini comme instance de la classe `time:Interval` (voir deuxième ligne du tableau 6.6), les deux instants désignés par les propriétés `time:hasBeginning` et `time:hasEnd` doivent être spécifiés de façon complète (année, mois, jour, heure, minute, seconde). Si ce n'est pas le cas, la traduction dans ONTOAST est complétée par des valeurs par défaut.

Comme dans le cas des informations spatiales, nous définissons des *tags* pour les variables temporelles obtenues suite à la traduction d'une ontologie OWL 2/ OWL vers ONTOAST. Ils sont utilisées pour guider la traduction des informations temporelles dans le sens inverse, depuis ONTOAST vers OWL 2/OWL, de façon analogue aux tags spatiaux.

Description temporelle définie par rapport à OWL-Time	Type AROM-ST	Tag
`time:Instant` + `time:inXSDDateTime`	instant	"instant"
`time:Interval` + `time:hasBeginning` + `time:hasEnd` + `time:Instant`*	interval	"interval"
`time:DurationDescription`	duration	"duration"
`time:DateTimeInterval` + `time:hasDateTimeDescription` + `time:DateTimeDescription`*	interval	"dateTime1"
`time:DateTimeDescription` + `time:year` + `time:month` +`time:day`+ `time:hour` + `time:minute`	instant	"dateTime2"
`time:DateTimeDescription` + `time:xsdDateTime`	instant	"dateTime3"
`time:Instant` + `time:inDateTime`+ `time:DateTimeDescription`*	instant	"inDateTime"

TAB. *6.6: Traduction dans ONTOAST des données temporelles décrites par rapport à l'ontologie OWL-Time.*

Après avoir passé en revue les correspondances qui existent entre, d'une côté, les types spatiaux et temporels proposés par AROM-ST, et, les concepts et relations définis dans les ontologies de référence *GeoRSS-Simple* et *OWL-Time*, de l'autre, nous présentons notre approche de défini-

tion et de gestion des relations spatiales et temporelles qualitatives.

6.4 Modèle spatial qualitatif en ONTOAST

Dans cette section, nous proposons une extension spatiale qualitative de AROM-ONTO. Celle-ci cherche à faire coexister, à la fois des données spatiales et temporelles quantitatives sous la forme de *géométries* ou d'*instants* et d'*intervalles* de temps, et des données imprécises sous la forme de relations spatiales et temporelles qualitatives. Nous prenons en compte trois catégories de relations spatiales qualitatives : *topologiques*, d'*orientation* et de *distance*, qui peuvent être issues d'une déclaration explicite de la part de l'utilisateur, ou bien inférées à partir d'informations existantes. Ainsi, lors d'une requête utilisateur, la réponse sera construite :

a) en utilisant les connaissances explicites, dans le cas où la relation cherchée est stockée dans la base ;

b) en réalisant des inférences qualitatives à partir des relations explicites existant entre les objets ;

c) en utilisant des méthodes numériques d'estimation et de calcul pour déduire des relations qualitatives impliquant les individus visés par la requête ;

d) en appliquant des raisonnements qualitatifs autant sur les relations spatiales descriptives existantes, que sur celles déduites par l'intermédiaire des calculs numériques.

Pour répondre à ces besoins, notre système de raisonnement spatial et temporel doit :

– définir à travers une extension du LMA, des opérateurs primitifs pour la déduction de relations spatiales et temporelles qualitatives, à partir de données quantitatives vectorielles. Ces primitives doivent être capables de calculer ou de vérifier si une relation R est respectée par une paire d'objets (x,y). Autrement dit, pour un x et un y donnés, il faut vérifier si $R(x,y)$ existe.
– fournir une définition pour toutes les relations spatiales et temporelles qualitatives prises en compte (i.e. `Disjoint`, `Intersects`, `NorthOf`, `During`, *etc.*).
– mettre à disposition des mécanismes d'inférence de nouvelles relations à partir des relations connues, en exploitant les caractéristiques, telles que la symétrie, la transitivité, *etc.*, des relations spatiales et temporelles qualitatives, et/ou les tableaux de composition (cf. sections 2.3.2.2, 2.3.2.5 et 2.3.2.7).

6.4.1 Définition du méta-modèle spatial

Nous avons intégré dans ONTOAST un modèle spatial qualitatif prédéfini contenant trois types de relations spatiales qualitatives : les relations topologiques, les relations de direction (ou d'orientation) et les relations de distance. La structure, le comportement et la sémantique de ces types de relations sont définis respectivement par les méta-classes `AROMTopology`, `AROMDirection` et `AROMDistance` (voir figure 6.9). Celles-ci sont des spécialisations de la méta-classe `AROMQSR` (de Arom Qualitative Spatial Relation), utilisée pour définir de façon générique une relation spatiale qualitative comme étant une association binaire dont les deux rôles (`subject` et `object`) désignent des objets spatiaux. Dans notre modèle, un objet spatial est une instance de la méta-classe `AROMSpatialConcept`. Comme illustré dans la figure 6.9, la méta-classe `AROMSpatialConcept` définit l'ensemble d'objets ayant une dimension spatiale, modélisée par une variable dont le type

est l'un des types spatiaux définis par AROM-ST. La méta-classe `SpatialType` est introduite exclusivement pour des raisons d'illustration, pour désigner l'ensemble des types spatiaux présentés dans la section 6.3.1 (voir figure 6.2). Il est important de souligner le fait que la géométrie d'un objet spatial peut ne pas être précisée.

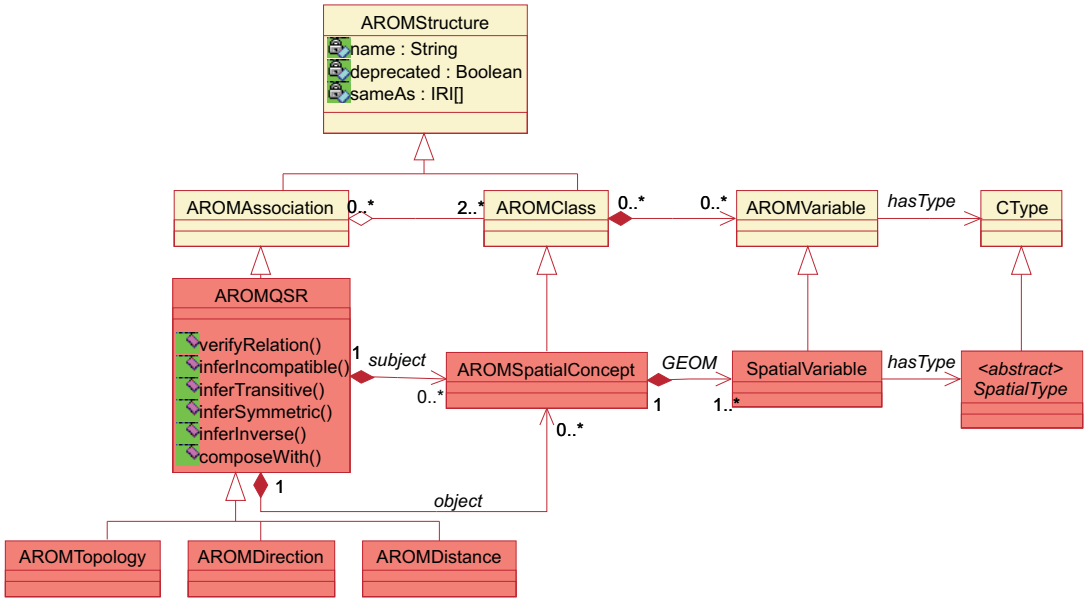


FIG. *6.9: La définition de la méta-classe **AROMQSR** et de ses trois spécialisations, **AROMTopology**, **AROMDirection** et **AROMDistance**, qui modélisent les relations spatiales qualitatives en ONTOAST.*

La méta-classe `AROMTopology` précise les caractéristiques générales des huit relations topologiques prises en compte par ONTOAST et définies par le calcul RCC-8 (voir section 2.3.2.1). Il s'agit des relations `disjoint`, `touches`, `inside`, `contains`, `equals`, `tangent_contains`, `tangent_inside` et `overlaps`.

Pour représenter les relations de direction, nous avons recours à des associations binaires (modélisées par la méta-classe `AROMDirection`) qui modélisent les 31 relations de direction proposées par l'approche *CDR* [224], pour deux régions de type REG^* (`North`, `East`, `South`, `West`, `North:West`, `West:South:East`, ...).

La méta-classe `AROMDistance`, modélise des relations quaternaires, qui précisent, en plus des deux objets spatiaux (l'objet cible et l'objet de référence), une métrique qui définit le type de distance employée et une échelle par rapport à laquelle est définie la relation de distance. Nous avons pris en compte trois relations de distance : `veryClose`, `close` et `far`.

6.4.1.1 Relations topologiques en ONTOAST

Afin de modéliser en ONTOAST des relations topologiques entre des objets spatiaux étendus, représentés par des polygones, nous avons défini un modèle spatial qualitatif. Celui-ci contient les relations, illustrées dans la figure 6.10, qui correspondent aux relations du calcul RCC-8 (présentées dans la section 2.3.2.1) comme suit : `disjoint` ≡ `DC`, `touches` ≡ `EC`, `contains` ≡ `NTPP`, `inside` ≡ `NTPPi`, `equals` ≡ `EQ`, `tangent_contains` ≡ `TPP`, `tangent_inside` ≡ `TPPi`, `overlap` ≡ `PO`. Nous avons choisi de ne pas utiliser les symboles des primitives RCC-8, car ils ne sont pas très explicites pour les utilisateurs non experts. En échange, nous nous sommes inspirés de

l'ensemble de relations proposées par l'ontologie `SpatialRelation`[1] développée par Ordnance Survey (voir section 3.3.2).

À ces relations, nous avons ajouté une relation générique, `spatiallyRelated`, qui représente tous les types de relations topologiques que nous avons considérés. L'introduction dans le modèle topologique de cette relation a la motivation suivante : si la relation topologique satisfaite par deux objets spatiaux *x* et *y* n'est pas connue, il est au moins possible de relier *x* et *y* par un tuple de type `spatiallyRelated`. Au fur et à mesure de l'ajout d'information sur les objets *x* et *y*, le processus de classification d'instance peut faire migrer ce tuple vers l'une des associations plus spécifiques.

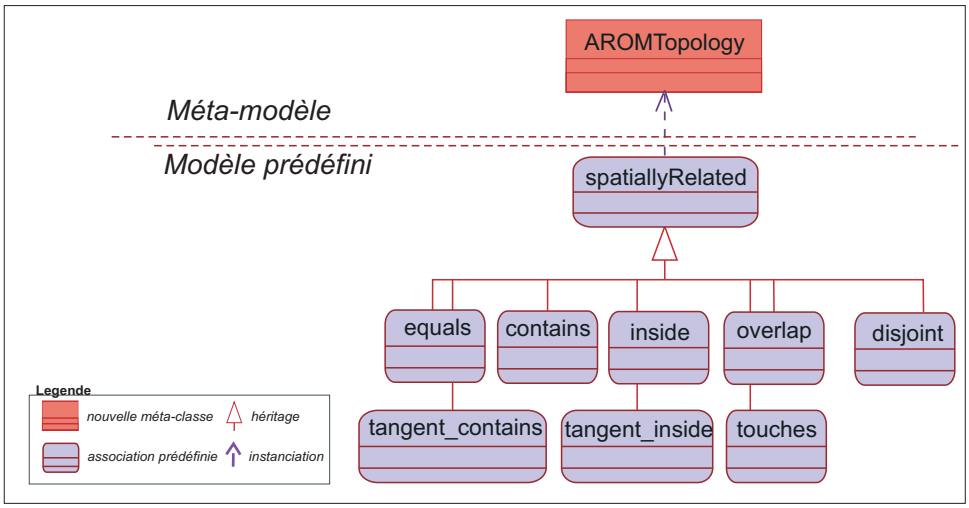


FIG. *6.10: Les relations topologiques prédéfinies ajoutées au modèle d'ONTOAST.*

Les relations topologiques sont définies pour des objets représentés par des polygones, mais peuvent également être appliquées à des géométries moins complexes (`line` ou `point`). Néanmoins, lors de la comparaison de la position relative de deux points, par exemple, seules deux relations topologiques ont du sens : `equals` et `disjoint`. Le tableau 6.7 illustre les relations topologiques qui peuvent être obtenues entre deux objets ayant des géométries à différents niveaux de complexité.

Pour les relations topologiques illustrées dans la figure 6.10, nous fixons la sémantique ainsi que le comportement. Cela veut dire qu'elles peuvent être instanciées sans redéfinition dans toute base de connaissances. Par exemple, pour préciser que la relation `contains` est valide entre deux objets spatiaux, *Paris* et *currentPos*, il suffit de définir un tuple de la relation `contains` comme illustré dans la figure 6.11.

[1] http ://www.ordnancesurvey.co.uk/oswebsite/ontology/SpatialRelations/v0.2/SpatialRelations.owl

Type de géométrie (*objet cible*)	Type de géométrie (*objet de reference*)	Relations topologiques valides
`polygon`	`polygon`	`equals`, `contains`, `inside`, `overlap`, `tangent_contains`, `tangent_inside`, `touches`, `disjoint`
`polygon`	`polyline`	`contains`, `overlap`, `tangent_contains`, `touches`, `disjoint`
`polygon`	`point`	`contains`, `touches`, `disjoint`
`polyline`	`polygon`	`inside`, `overlap`, `tangent_inside`, `touches`, `disjoint`
`polyline`	`polyline`	`touches`, `equals`, `disjoint`
`polyline`	`point`	`touches`, `disjoint`
`point`	`polygon`	`inside`, `touches`, `disjoit`
`point`	`polyline`	`touches`, `disjoint`
`point`	`point`	`equals`, `disjoint`

TAB. *6.7: Les relations topologiques qui s'appliquent entre différents types de géométries.*

```
instance: Paris
   is-a:City
instance: currentPos
   is-a: GPSPos
   pos=#48.8566667 2.3509871#
tuple:
   is-a: contains
   subject=Paris
   object=currentPos
```

FIG. *6.11: Définition en ONTOAST de la relation topologique d'inclusion entre la ville de Paris et la position courante de l'utilisateur (currentPos).*

6.4.1.2 Relations de direction en ONTOAST

Les relations de direction que nous avons définies en ONTOAST correspondent aux relations proposées par le modèle *CDR*[2] [224], présenté dans la section 2.3.2.4. Concrètement, nous considérons deux types de relations définies pour des régions de type REG^* (voir notation 2.1, page 31) : les relations de type *carreau unique* et les relations composées, de type *carreaux multiples* (les deux définies par la méta-classe `AROMDirection`) (voir figure 6.12). Les relations de direction sont définies comme des associations binaires, ayant deux rôles prédéfinis : `subject` et `object` qui désignent respectivement l'*objet cible* et l'*objet de référence.*

Dans le modèle ONTOAST, une relation de direction entre deux objets *A* et *B*, est définie

[2]Nous rappelons au lecteur que *CDR* est une méthode de modélisation des relations de direction binaires entre objets spatiaux étendues (des régions), qui utilise l'approximation de l'*objet de référence* par son rectangle englobant minimal pour partitionner l'espace en quatre carreaux de direction de type trapèze et un carreau de type rectangle. Pour trouver la relation de direction entre un *objet cible* et l'*objet de référence*, on considère les carreaux qui sont croisés par la géométrie de l'objet cible.

comme un tuple d'une des associations prédéfinies `North`, `West`, `North:South`, `North:West:East` ...

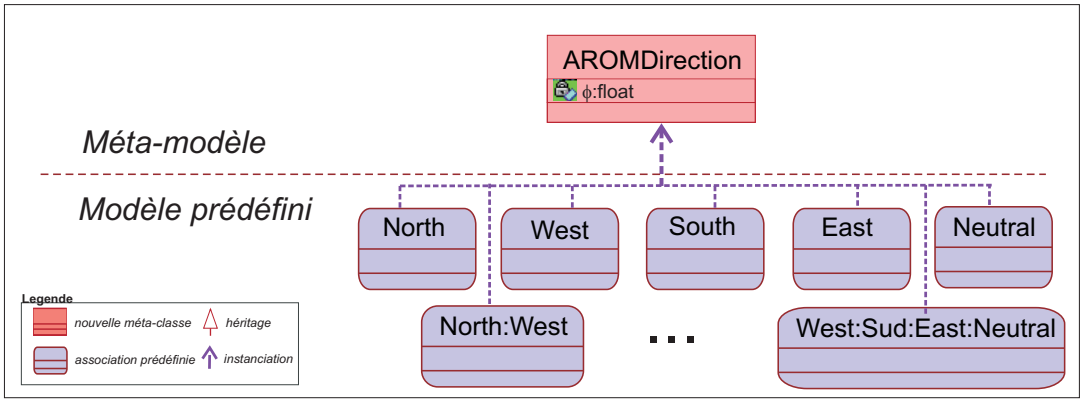


FIG. *6.12: Le modèle spatial prédéfini utilisé pour décrire des relations de direction en ONTOAST.*

Les relations de direction sont définies pour des objets représentés par des polygones, mais peuvent être adaptées pour des géométries moins complexes (ligne ou point). Néanmoins, lorsqu'on cherche la relation de direction entre deux points par exemple, l'ensemble des relations possibles est réduit à {`North`, `West`, `South`, `East`, `Neutral`}. Dans ce cas, les seules relations pertinentes sont les relations de type *carreau unique*. Le tableau 6.8[3] illustre les relations de direction qui peuvent être obtenues entre deux objets ayant des géométries à différents niveaux de complexité.

Type de géométrie (*objet cible*)	Type de géométrie (*objet de référence*)	Relations de direction valides
`polyline` ou `polygon`	`polyline` ou `polygon`	δ(`North`,`West`,`South`,`East`,`Neutral`)
`polyline` ou `polygon` ou `point`	`point`	`North`, `West`, `South`, `East`, `Neutral`
`point`	`polyline` ou `polygon`	δ(`North`,`West`,`South`,`East`)

TAB. *6.8: Les relations de direction qui s'appliquent entre différents types de géométrie.*

Pour les relations de direction illustrées dans la figure 6.12, nous avons prédéfini la sémantique ainsi que le comportement. Cela veut dire qu'elles peuvent être instanciées sans redéfinition dans toute base de connaissances. Par exemple, pour définir une relation `North:East:Neutral` entre la *Roumanie* et l'*Ukraine*, il suffit de définir un tuple de la relation homonyme comme montré dans la figure 6.13.

Dans le cas où, pour une paire d'objets spatiaux (A,B), la base de connaissances BC contient plusieurs tuples de relations de direction, $R_1,\ldots,R_k$, alors on considère que la relation de direction entre A et B est la disjonction $\cup_{i=1}^{k} R_i$.

[3]La fonction δ désigne l'ensemble des relations de direction qui peuvent être construites à partir des relations élémentaires qu'elle reçoit comme paramètres (voir la définition de δ donnée dans la section 2.3.2.5).

```
instance: Romania
   is-a:Country
instance: Ukraine
   is-a: Country
tuple:
   is-a: North:East:Neutral
   subject=Romania
   object=Ukraine
```

FIG. *6.13: Exemple de relation de direction de type carreaux multiples définie en ONTOAST.*

6.4.1.3 Relations de distance en ONTOAST

Nous considérons l'existence de trois relations de distance : `veryClose`, `close` et `far`, qui décrivent le fait qu'un objet *A* est respectivement *très proche*, *proche* et *loin* d'un deuxième objet, *B*, dit de référence. Les relations de distance sont définies en ONTOAST entre des régions de type REG^*. Elles sont modélisées comme des associations quaternaires, qui spécifient, en plus des deux objets spatiaux *A* et *B*, une métrique et une échelle par rapport auxquelles la distance est définie (voir figure 6.14).

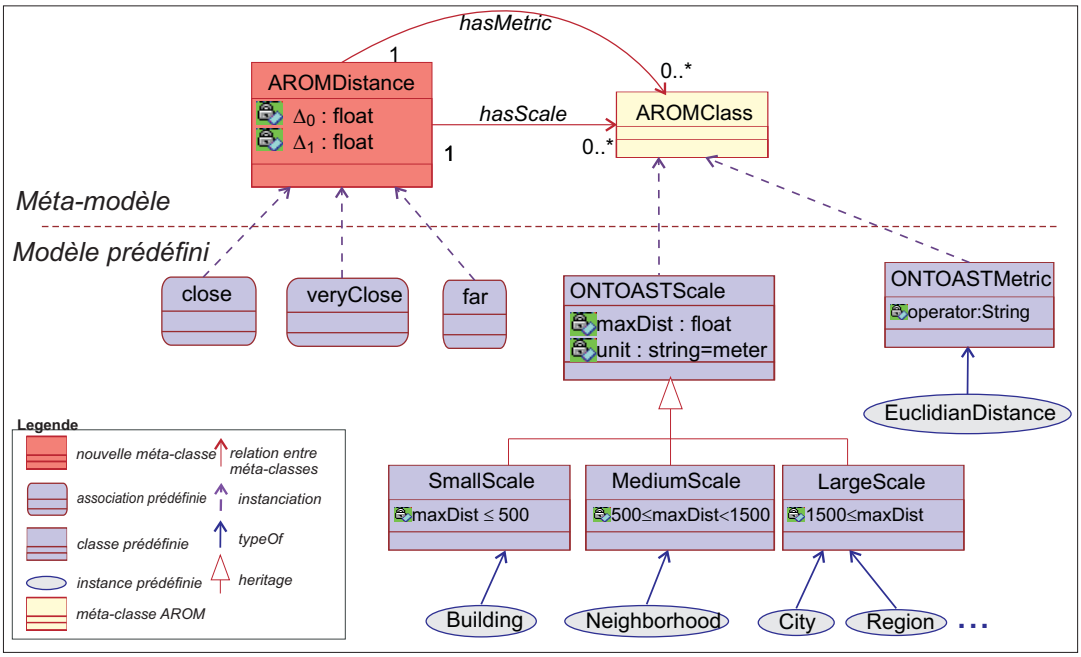


FIG. *6.14: Les relations de distance ajoutées au modèle spatial prédéfini d'ONTOAST.*

Pour définir l'échelle utilisée lors de la spécification des relations de distance, nous avons suivi l'exemple de Gärling et Golledge [112] qui considèrent trois types d'"environnements" : *i)* à petite échelle (*i.e.* une chambre ou un appartement), *ii)* à échelle moyenne (*i.e.* un bâtiment ou un quartier) et *iii)* à grande échelle (*i.e.* des villes, régions, pays, *etc.*). Une approche similaire est présentée dans [177] mais les échelles choisies sont rapportées aux dimensions du corps humain. Ainsi, quatre catégories d'espace sont identifiées par Montello [177] : *figural*, *vista*, *environnemental* et *géographique*. L'espace *figural* peut être perçu en entier sans déplacement. C'est le cas des objets qui se trouvent sur une table, les éléments d'une photo ou d'un tableau *etc.* dès lors

qu'ils sont dans le champ de vision de la personne. L'espace *vista* fait référence à un espace à petite échelle. L'espace *environnemental* est plus grand que le corps et l'entoure. Il est en effet trop grand et autrement obscurci pour être appréhendé directement sans déplacement. C'est l'espace des bâtiments, les quartiers et les villes et peut être associé à l'*échelle moyenne* dans notre framework. L'espace *géographique* est beaucoup plus grand que le corps humain et ne peut pas être appréhendé directement par la locomotion. Il doit plutôt être appris par l'intermédiaire de représentations symboliques telles que des cartes ou à travers des modèles qui réduisent essentiellement l'espace géographique à une représentation figurative [177].

Dans ONTOAST, l'échelle est modélisée par la classe prédéfinie `ONTOASTScale`, dont la définition est inspirée des travaux évoqués auparavant, notamment ceux de Gärling et Golledge [112]. Ainsi, la classe `ONTOASTScale` dispose de trois spécialisations, `SmallScale`, `MediumScale` et `LargeScale` qui décrivent des espaces à petite échelle, à échelle moyenne et à grande échelle. Elles sont caractérisées par la valeur de la variable `maxDist` qui donne la valeur maximale de la distance dans l'espace considéré. Par exemple, si la variable `distMax` a la valeur 500 m, alors on ne peut pas avoir une distance supérieure à 500 mètres qui soit raportée à cette échelle.

Les relations de distance que nous avons introduites en ONTOAST, `veryClose`, `close` et `far` sont définies en fonction de la valeur de l'attribut `distMax` de l'échelle qui leur est associée. Les plages d'acceptation pour les relations de distance sont construites en fonction de la valeur de `distMax`, comme suit :

$$\Delta_0 + \Delta_1 + \Delta_2 = \texttt{distMax} \tag{6.1}$$

$$\Delta_1 = 3 * \Delta_0 \tag{6.2}$$

$$\Delta_2 = 6 * \Delta_0. \tag{6.3}$$

La plage d'acceptation de la relation `veryClose` est $\delta_0 = [0, \Delta_0]$, la plage d'acceptation de la relation `close` est $\delta_1 = (\Delta_0, 4 * \Delta_0]$ et la plage d'acceptation pour la relation `far` est $\delta_2 = (4 * \Delta_0, \texttt{distMax}]$.

Par exemple, si on définit, pour une paire d'objets *A* et *B*, une relation `veryClose` par rapport à l'échelle `City`, pour laquelle la variable `distMax` a la valeur 50.000 m, alors la plage d'acceptation pour la distance `veryClose` est calculée selon l'algorithme suivant : $\Delta_0 + \Delta_1 + \Delta_2 = 50.000$. En sachant que nous avons choisi des plages de distance monotones croissantes, qui respectent les restrictions 6.2 et 6.3, on peut déduire que $\Delta_0 = 50.000/10 = 5.000$. Donc à l'échelle `City`, seuls les objets spatiaux qui se trouvent à moins de cinq kilomètres sont considérés comme étant très proches. Les objets qui se trouvent entre 5 km et 15 km sont considérés proches et tous les autres sont considérés loin.

D'autres échelles peuvent être définies si besoin, comme instances de la classe `ONTOASTScale` avec des valeurs personnalisées pour le paramètre `distMax`. Une étude sur la représentation spatiale à différentes échelle peut être consultée dans [212]. De plus, le calcul des attributs Δ_0 et Δ_1 et en conséquence la définition même des relations `close`, `veryClose` et `far`, peut également être adapté aux particularités du domaine d'application. Cela est fait à travers la spécification d'une nouvelle équation de calcul de Δ_0 et Δ_1 au niveau de la définition des associations ONTOAST prédéfinies `close`, `veryClose` et `far`.

En ce qui concerne la métrique, nous avons défini un seul type de distance, la distance Euclidienne (voir figure 6.14), calculée entre les centres des rectangles minimales englobants des objets *A* et *B*. La valeur de la distance est calculée en utilisant l'opérateur LMA `distance`. D'autres opérateurs pour le calcul de la distance peuvent être aisément ajoutés au LMA. On peut donc envisager l'utilisation d'autres types de distance si besoin.

Pour les relations de distance illustrées dans la figure 6.14, nous avons fixé leur sémantique ainsi que leur comportement. Cela veut dire qu'elles peuvent être instanciées sans redéfinition dans toute base de connaissances. Un exemple de définition d'une relation `veryClose` entre une position GPS et la tour Eiffel, est donné dans la figure 6.15.

```
instance: EiffelTower
   is-a:Monument
instance: currentPos
   is-a: GPSPos
   pos=#48.8566667 2.3509871#
tuple:
   is-a: veryClose
   subject=currentPos
   object=EiffelTower
   hasScale=City
   hasMetric=EuclidianDistance
```

FIG. *6.15: Exemple de relation de distance définie en ONTOAST.*

6.4.2 Algorithme de déduction des relations spatiales

Afin de déduire la relation topologique, de direction ou de distance qui existe entre deux objets spatiaux, *A* et *B*, décrits dans une base de connaissances ONTOAST, *BC*, nous avons défini l'algorithme générique `SpatialRelations`. Celui-ci présente peu de différences selon qu'il est utilisé pour la déduction de l'un ou l'autre des trois types de relations spatiales gérées par ONTOAST. Il prend comme paramètres d'entrée la base de connaissances de référence, *BC*, les deux objets spatiaux, *A* et *B*, pour lesquels on cherche à établir la relation spatiale et deux ensembles $\mathcal{D}$ et $\mathcal{D}^*$ qui varient d'un type de relation à un autre comme suit :

- pour les relations topologies, $\mathcal{D} = \mathcal{T}$, où $\mathcal{T} =_{def} \{\texttt{equals}, \texttt{contains}, \texttt{inside}, \texttt{overlap}, \texttt{tangent\_contains}, \texttt{tangent\_inside}, \texttt{touches}, \texttt{disjoint}\}$, $\mathcal{D}^* = 2^{\mathcal{T}}$
- pour les relations de direction, $\mathcal{D} = \mathcal{D}ir$, où $\mathcal{D}ir = \delta(\texttt{North}, \texttt{West}, \texttt{South}, \texttt{East}, \texttt{Neutral})$[4], $\mathcal{D}^* = 2^{\mathcal{D}ir}$ et
- pour les relations de distance $\mathcal{D} = \Lambda$, où $\Lambda = \{\texttt{veryClose}, \texttt{close}, \texttt{far}\}$, $\mathcal{D}^* = 2^{\Lambda}$.

L'algorithme utilise à la fois les connaissances explicites contenues dans la base *BC* et des techniques de raisonnement spatial quantitatif et qualitatif. Il considère des objets spatiaux représentés par des polygones, mais peut être facilement adapté pour traiter des géométries moins complexes.

Par la suite, nous expliquons l'utilisation de l'algorithme `SpatialRelations` pour la déduction des relations topologiques, en sachant que celui-ci s'applique de façon analogue pour la déduction des relations de direction et de distance, à l'exception de quelques modifications ponctuelles qui doivent être mises en place et que nous décrivons dans les sections 6.4.2.2. et 6.4.2.3.

[4] La fonction δ désigne l'ensemble des relations de direction qui peuvent être construites à partir des relations élémentaires qu'elle reçoit comme paramètres (voir la définition de δ donnée dans la section 2.3.2.5).

6.4.2.1 Algorithme `SpatialRelations` pour la déduction des relations topologiques

Dans un premier temps, le système vérifie si, dans la base de connaissances *BC*, le tuple (A,B) est défini comme appartenant à l'extension d'une relation topologique élémentaire (voir lignes 1 à 4). Si c'est le cas, l'algorithme termine car la solution a été trouvée. Dans le cas où la relation entre *A* et *B* n'est pas explicite, le système cherche à la déduire à partir de sa relation inverse si celle-ci est explicite dans la base de connaissances *BC* (voir lignes 6 à 9). Concrètement, si la base de connaissances *BC* contient un tuple d'une relation topologique élémentaire, défini comme reliant les objets *B* et *A* alors le système déduit la relation entre *A* et *B* à partir du tableau des relations inverses (voir tableau 6.9). Cette opération est réalisée par la fonction $inv(R)$.

Relation topologique	Inverse
`disjoint`	`disjoint`
`equals`	`equals`
`contains`	`inside`
`tangent_contains`	`tangent_inside`
`overlap`	`overlap`
`touches`	`touches`
`inside`	`contains`
`tangent_inside`	`tangent_contains`

TAB. *6.9: Les relations topologiques de RCC8 et leurs inverses.*

Si cela n'est pas possible, parce que la relation entre *B* et *A* n'est pas explicite, ou qu'elle n'est pas élémentaire, le système vérifie si la base de connaissances *BC* contient un objet spatial *C* tel que les relations topologiques entre *A* et *C* et entre *B* et *C* soit explicites et élémentaires. Si c'est le cas, en utilisant le tableau de composition 2.6 le système déduit l'ensemble de relations possibles entre *A* et *B* (voir lignes 11 à 15). Il faut noter que le résultat de la composition, calculée par la fonction $compose(R_1,R_2)$, peut être une relation élémentaire, mais également une disjonction de relations élémentaires.

Si la relation topologique entre *A* et *B* n'a pas pu être déduite avec les trois premiers tests, le système vérifie à l'aide de la fonction *knownGeom* si les géométries des objets *A* et *B* sont disponibles dans la base de connaissances *BC* (voir lignes 17 à 20). Si c'est le cas, en utilisant les opérateurs topologiques du LMA, le système calcule la relation topologique entre *A* et *B*. La méthode de calcul utilisée est décrite dans la section 6.4.3.1.

Si, au contraire, les deux géométries ne sont pas disponibles, le système construit à partir des relations spatiales décrites dans la base de connaissances *BC* un réseaux de contraintes, *CtNetwork*, qui sera envoyé à un *solver* ECL^iPS^e[5] [21](voir la description de la méthode de définition du réseau et de calcul des relations topologiques dans la section 6.4.4.1). En imposant l'arc-consistance généralisée (voir définition 6.1) pour le réseau de contraintes *CTNetwork*, le

[5] ECL^iPS^e est un framework pour la programmation par contraintes. Il intègre un solveur linéaire, un solveur non-linéaire, une interface pour des solveurs externes à base de Simplexe (*eplex*), un solveur sur les domaines finis (*ic*) et d'autres solveurs distribués sous formes de librairies.

solveur affine autant que possible les relations topologiques entre les régions considérées.

Données: BC - base de connaissances ONTOAST
Entrées: A, B objets spatiaux définis dans BC
Sorties: $Rel \subseteq \mathcal{D}^*$ - la relation **topologique**, de **distance** ou de **direction** la plus spécifique qui peut être déduite ou calculée entre les régions A et B à partir des connaissances contenues dans la base BC

si $\exists! R \in \mathcal{D} : R(A,B) \in BC$ // la relation entre A et B est explicite et élémentaire
alors
  $Rel \leftarrow R$
  Return Rel
fin
si $\exists! R \in \mathcal{D} : R(B,A) \in BC$ // la relation entre B et A est explicite et élémentaire
alors
  $Rel \leftarrow inv(R)$
  Return Rel
fin
si $\exists! R_1, R_2 \in \mathcal{D} : R_1(A,C) \in BC \wedge R_2(C,B) \in BC$ // la relation entre B et A peut être déduite à travers la composition des relations $R_1(A,C)$ et $R_2(C,B)$
alors
  $Rel \leftarrow compose(R_1, R_2)$
  $BC \leftarrow BC \cup Rel(A,B)$// les relations dont l'inférence n'est pas élémentaire sont ajoutées dans la base BC
  Return Rel
fin
si $knownGeom(A) \wedge knownGeom(B)$ **alors**
  $Rel \leftarrow computeRelation(Geom(A), Geom(B))$
  $BC \leftarrow BC \cup Rel(A,B)$
  Return Rel
fin
$CtNetwork \leftarrow buildCtNetwork(BC, spatialRelation)$// le paramètre *SpatialRelations* spécifie le type de réseaux de contraintes qu'on veut construire : *topologique*, *directionnel*, de *distance* or *mixte*
$ArcConsistency(CtNetwork)$
$update(BC, CtNetwork)$
si $\exists R \in \mathcal{D} : R(A,B) \in BC$ **alors**
  $Rel \leftarrow \cup_{R \in \mathcal{D} : R(A,B) \in BC} R$ // $Rel \in \mathcal{D}^*$
  Return Rel
fin
Return $\emptyset$ // pas de solution

Algorithme 1: SpatialRelation

La base de connaissances sera ensuite mise à jour pour refléter les inférences réalisées. Il faut

noter que, pour les paires de régions pour lesquelles le *solveur* n'a pas réussi à déduire une relation topologique plus spécialisée que `spatiallyRelated`, aucun tuple de relation topologique ne sera ajouté à BC.

Le dernier test de l'algorithme `SpatialRelations` (lignes 25 à 29) vérifie si, suite à l'utilisation du réseau de contraintes *CTNetwork*, une relation topologique entre A et B a pu être déduite. Si c'est le cas, la relation résultat *Res* est construite comme disjonction des relations topologiques contenues par la base BC mise à jour, pour les objets A et B. Si ce n'est pas le cas, le résultat est l'ensemble vide.

6.4.2.2 Algorithme `SpatialRelations` pour la déduction des relations de direction

L'algorithme `SpatialRelations` s'applique de façon analogue pour la déduction de la relation de direction entre les objets spatiaux A et B. La principale adaptation de l'algorithme à la spécificité des relations de direction concerne la définition des fonctions *inv*, *compose* et *computeRelation*. Ainsi, pour trouver l'inverse d'une relation de direction on utilise la théorème 2.3 (page 36). Pour la déduction de la relation de direction entre A et B à travers la composition (voir lignes 11 à 15 de l'algorithme `SpatialRelations`) on utilise les algorithmes de composition présentées en [224]. Finalement, pour le calcul de la relation de direction à partir des géométries des objets A et B on utilise l'opérateur du LMA `direction`, décrit dans la section 6.4.3.2.

6.4.2.3 Algorithme `SpatialRelations` pour la déduction des relations de distance

Pour utiliser l'algorithme `SpatialRelations` pour la déduction des relations de distance, plusieurs modifications doivent être prises en compte. Dans un premier temps, pour déduire la relation de distance entre A et B à partir de la relation de distance qui existe entre B et A, il faut s'assurer que la distance considérée est symétrique. Ainsi, l'algorithme `SpatialRelations` adapté pour les relations de distance prend un paramètre d'entrée en plus : la métrique M utilisée pour calculer la distance. De plus, les lignes 6 à 9 sont remplacées par la séquence qui suit :

```
1 si symmetric(M) and ∃!R ∈ 𝒟 : R(B,A) ∈ BC alors
2 |   Rel ← R// la relation de distance est symétrique
3 |   Return Rel
4 fin
```

Si la relation de distance entre A et B n'est pas explicite, et ne peut pas être obtenue à partir de son inverse, on continue le raisonnement en cherchant à déduire la relation de distance entre A et B à travers la composition. Concrètement on cherche dans la base BC un objet spatial intermédiaire C tel que :

a) on connaît la relation de distance entre A et C et entre C et B et

b) la relation de direction D_1 entre A et C et la relation de direction D_2 entre C et B sont connues et respectent :

(a) $D_1 = D_2$ ou

(b) $D_1 = -D_2$

où $\forall D = R_1 : \ldots : R_k, R_i \in \{North, West, South, East, Neutral\}, i \in [1..k] : -D = R_1^{\downarrow} : \ldots : R_k^{\downarrow}$.

Dans le cas où les relations de direction D_1 et D_2 sont identiques (respectivement opposées) on utilise le tableau de composition 2.7 (respectivement 2.8) pour déduire la relation de distance entre A et B. Pour refléter ces tests, les lignes 11 à 15 de l'algorithme `SpatialRelations` doivent être remplacées par :

```
1  si ∃!R1,R2 ∈ 𝒟 : R1(A,C) ∈ BC ∧ R2(C,B) ∈ BC alors
2      si direction(A,C) = direction(C,B) alors
3          Rel ← compose1(R1,R2)// Rel ∈ 2^𝒟
4          BC ← BC ∪ Rel(A,B)// la relation résultat est ajoutée dans la base BC
5          Return Rel
6      fin
7      sinon si direction(A,C) = opposite(direction(B,C)) alors
8          Rel ← compose2(R1,R2)
9          BC ← BC ∪ Rel(A,B)
10         Return Rel
11     fin
12 fin
```

Dans le cas où la distance ne peut pas être déduite à travers la composition, la prochaine étape de l'algorithme consiste à vérifier si les informations quantitatives sur la géométrie des objets A et B sont disponibles (voir lignes 17 à 20 de l'algorithme `SpatialRelations`). Si c'est le cas, la relation de distance peut être déduite à travers des calculs mathématiques présentes dans la section 6.4.3.3.

6.4.3 Calcul des relations spatiales à partir des géometries

6.4.3.1 Calcul des relations topologiques à partir des données spatiales quantitatives

Une relation topologique peut être explicitement définie par l'utilisateur, mais également calculée. Pour calculer une relation topologique à partir des données quantitatives, nous utilisons les opérateurs géométriques définis par AROM-ST sur les géométries des objets. Nous avons ajouté au LMA un nouvel opérateur, `geom` qui renvoie la géométrie attachée à un objet spatial donné comme paramètre (instance de `AROMSpatialConcept`). Nous avons également défini deux autres opérateurs topologiques : `touches` et `tangent`. Ainsi, pour déterminer la relation topologique qui est satisfaite par deux objets spatiaux A et B, nous utilisons l'arbre de décision présenté dans la figure 6.16.

Suite à ce calcul, un tuple de la relation inférée reliant A et B sera ajouté dans la partie assertionelle (ABox) de la base BC. Pour les domaines où une distribution des relations topologiques est connue ou peut être déduite, on peut utiliser l'approche décrite dans [63] pour optimiser la structure de l'arbre de décision 6.16. Par exemple, pour une application dans le domaine du cadastre, sachant que 95% des relations topologiques sont de type `disjoint` et 5% de type `touches`, les auteurs utilisent exclusivement les deux opérateurs associés pour vérifier la relation entre deux parcelles P_1 et P_2.

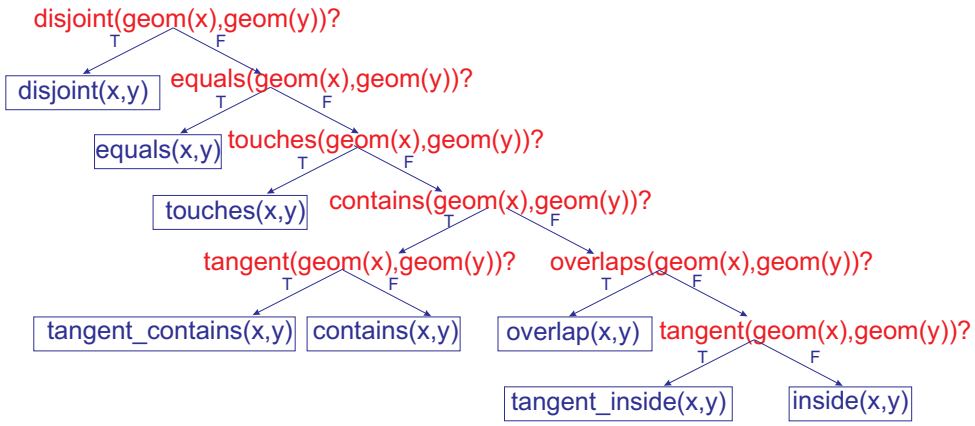


FIG. *6.16: Arbre de déduction de la relation topologique satisfaite par deux objets. En non encadré les opérateurs géométriques utilisés et en encadré les relations spatiales qualitatives déduites.*

6.4.3.2 Calcul des relations de direction à partir des données spatiales quantitatives

Afin de calculer la relation de direction que l'objet spatial *A* respecte par rapport à l'objet spatial *B*, à partir des données quantitatives, nous avons défini l'opérateur `direction`, que nous avons ajouté à l'ensemble d'opérateurs du LMA d'ONTOAST. L'opérateur `direction` est construit comme une adaptation pour l'approche *CDR* de l'algorithme de calcul des relations d'orientation proposé par [222] pour l'approche *PDR* (voir section 2.3.2.4). L'algorithme `ComputeDirection` décrit les calculs réalisés par l'opérateur `direction` pour trouver la relation de direction entre deux régions représentées par des ensembles de polygones.

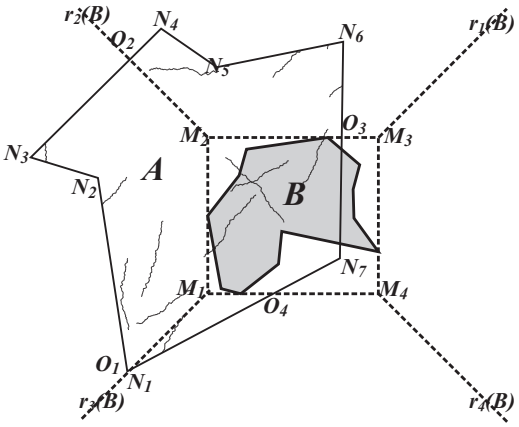


FIG. *6.17: Exemple de relation spatiale de type carreaux multiples définie en ONTOAST.*

L'algorithme considère deux régions *A* et *B*, représentées par les ensembles de polygones : $S_A = \{p_1, \ldots, p_k\}$ et, respectivement, $S_B = \{q_1, \ldots, q_l\}$. Afin de calculer la relation de direction entre *A* et *B*, il faut d'abord calculer le rectangle englobant minimal de la région *B* pour pouvoir construire les quatre carreaux extérieurs qui forment le système d'orientation. Ensuite, if faut vérifier quels sont les carreaux croisés par au moins un des polygones $p_1, \ldots, p_k$. Pour cela, Skiadopoulos *et al.*[222] proposent de diviser les arêtes de chaque polygone p_i de S_A de manière à ce que :

– la région *A* ne change pas ;

– chaque nouvelle arête est incluse dans un seul carreau.

Données: Deux ensembles de polygones S_A et S_B qui représentent les régions A et $B \in REG^*$.

Sorties: La relation de direction, *Rel*, tel que A *Rel* B est vrai.

1 $Rel \leftarrow \emptyset$
2 **pour** *chaque polygone p de* S_A **faire**
3 **pour** *chaque arête XY du polygone p* **faire**
4 **si** *X et Y sont inclus dans le même carreau T de B* **alors**
5 $R \leftarrow$ **tile-union**(R, T)
6 **fin**
7 **sinon**
8 Soit Γ l'ensemble de points d'intersection entre XY et les quatre segments de droite M_1M_2, M_2M_3, M_3M_4 et M_4M_1 et les quatre droites qui définissent les carreaux extérieures : $r_1(B)$, $r_2(B)$, $r_3(B)$ et $r_4(B)$
9 Soit $\{XO_1, \ldots, O_kY\}$ les segments obtenus suite à la division du XY à l'aide des points contenus par Γ
10 Replace XY par $\{XO_1, \ldots, O_kY\}$ dans la représentation du polygone p
11 Soit $T_1, \ldots, T_k$ les carreaux de B qui incluent les points médians des arêtes $\{XO_1, \ldots, O_kY\}$
12 $R \leftarrow$ **tile-union**$(R, T_1, \ldots, T_k)$
13 **fin**
14 **fin**
15 **si** *le centre de* ***mbb(b)*** *est dans p* **alors**
16 $R \leftarrow$ **tile-union**$(R, Neutral)$
17 **fin**
18 **fin**
19 Return *Rel*

Algorithme 2: ComputeDirection

À cette fin, pour chaque arête XY de la région A tel que X et Y se trouvent dans des carreaux différents, on calcule l'ensemble Γ de points d'intersection avec les segments qui forment le rectangle englobant minimal de B et avec les demi-droites qui définissent les carreaux extérieurs (ligne 7 de l'algorithme `ComputeDirection`). Pour la configuration spatiale illustrée dans la figure 6.17, on calcule, par exemple, les points d'intersection entre l'arête N_3N_4 et $[M_1M_2]$, $[M_2M_3]$, $[M_3M_4]$, $[M_4M_1]$ et les semi-droites $r_1(B)$, $r_2(B)$, $r_3(B)$ et $r_4(B)$. Le résultat est l'ensemble $\Gamma = \{O_2\}$, qui est ensuite utilisé pour diviser l'arête N_3N_4 en deux arêtes : N_3O_2 et O_2N_4. Chacune des deux arêtes ainsi obtenues est incluse dans un seul carreau et leur union est l'arête N_3N_4, donc elles peuvent être utilisées pour remplacer N_3N_4 dans la définition de A.

Cette segmentation est réalisée pour chaque arête de A qui traverse plusieurs carreaux de direction (voir lignes 7 à 11 de l'algorithme `ComputeDirection`), dans notre exemple N_3N_4, N_6N_7 et N_7N_1. Une fois la segmentation réalisée, pour trouver la relation de direction entre A et B, il suffit

de vérifier quels sont les carreaux de direction qui incluent au moins un segment de l'ensemble obtenu. Pour cela, on peut utiliser le point de milieu de chaque segment, parce que chaque arête se trouve dans le même carreau que son milieu (ligne 11). La relation de direction résultat est calculée à l'aide de la fonction `tile-union` qui combine les carreaux de B croisés par A. Dans le cas illustré par la figure 6.17, la relation obtenue est `North:West:South:Neutral`.

Dans le cas où, après segmentation, une arête XY, se trouve soit sur les demi-droites $r_1(B)$, $r_2(B)$, $r_3(B)$ et $r_4(B)$ soit sur les arêtes du *mbb(B)* (rectangle englobant minimal de B), elle est ignorée par l'algorithme. Cela est dû au fait que son point de milieu ne peut pas être utilisé pour déduire la position de l'arête dans un carreau de direction. Cependant, l'arête qui suit l'arête XY ou celle qui la précède peuvent supprimer l'incertitude. Néanmoins, si toutes les arêtes ont cette ambiguïté (*i.e.* $A = mbb(B)$), le dernier test réalisé par l'algorithme (ligne 14) sert à identifier la relation `Neutral` comme relation de direction entre A et B.

L'approche proposée par [222], que nous avons reprise et adaptée dans le cadre de cette thèse, a plusieurs avantages par rapport aux approches "classiques" de calcul de la direction entre deux régions, basées sur la *coupure des polygones*[6], et qui sont utilisées majoritairement dans le domaine des SIG et des bases de données spatiales. Les algorithmes de coupure des polygones par des rectangles fermées (*i.e.* rectangle englobant minimal) ou ouverts (*i.e.* un carreau de direction extérieur) sont très performants. Néanmoins, ils sont basés sur l'introduction de nouvelles arêtes dans le polygone cible p afin de pouvoir diviser p dans plusieurs polygones $p_1, \ldots, p_k$, tel que $p = p_1 \cup \ldots \cup p_k$ et que chaque polygone soit dans un seul carreau de direction. Dans certains cas, le nombre d'arêtes ajoutées peut être assez important [222]. De plus, pour pourvoir couper une région p, ses arêtes doivent être parcourues cinq fois (une fois pour chaque carreau de direction considéré). Dans des applications SIG réelles le nombre moyen d'arêtes est assez important, donc chaque parcours de l'ensemble d'arêtes peut être coûteux en temps [222]. Par ailleurs, les algorithmes de coupure de polygones impliquent aussi des opérations en virgule flottante complexes qui sont également coûteuses [222]. L'approche de Skiadopoulos *et al.* introduit beaucoup moins d'arêtes que la méthode de coupure des polygones et elle est réalisée en un seul parcours des arêtes. La complexité de l'algorithme est linéaire par rapport au nombre d'arêtes du polygone p.

6.4.3.3 Calcul des relations de distance à partir des données quantitatives

Afin de calculer la relation de distance entre deux objets spatiaux A et B à partir de leur géométries, nous utilisons l'opérateur LMA spécifié par la variable `operator` de la métrique associée, M. Dans le cas de la distance Euclidienne, l'opérateur employé est `distance`. Celui-ci calcule la distance Euclidienne, d, entre les deux points qui correspondent aux centres des rectangles englobants minimaux des objets A et B. La valeur numérique d est ensuite comparée aux paramètres Δ_1 et Δ_2, eux-mêmes dépendants de l'échelle par rapport à laquelle on définit la relation de distance. Ainsi, en utilisant les équations 6.1-6.3 (page 170), on calcule l'intervalle d'acceptation dans lequel s'inscrit la distance d, et on génère en conséquence la relation de distance qui correspond à cet intervalle.

[6] L'approche est connue sur le nom de *polygon clipping*.

6.4.4 Déduction des relations spatiales à travers le raisonnement qualitatif

Une autre façon de déduire une relation spatiale entre deux objets spatiaux consiste à utiliser le raisonnement qualitatif basé sur les définitions axiomatiques des relations spatiales présentées dans la section 2.3. Pour cela, nous nous sommes basés sur le travail de Brand, dont les résultats sont présentés dans [41, 42]. L'auteur propose d'aborder le raisonnement spatial comme un problème de satisfaction de contraintes (CSP[7]). L'originalité de ce travail vient du fait que les relations spatiales sont modélisées comme des variables et non pas comme des contraintes.

Problème de satisfaction de contraintes (ou CSP)

Pour introduire les concepts principaux du domaine de la satisfaction des contraintes, considérons une séquence $X = \{x_1, \ldots, x_n\}$ de variables disjointes deux à deux, et ayant les domaines de valeurs $D_1, \ldots, D_n$. La notion de contrainte C sur X, $C(X)$, désigne un sous ensemble de $D_1 \times \ldots \times D_n$.

Un problème de satisfaction de contraintes est défini comme étant un triplet $\langle X, D, C \rangle$ où $C = \{c_1, \ldots, c_m\}$, et chaque c_j est une contrainte de *CSP* tel que $var(c_j) = \{x_{j1}, \ldots, x_{jk}\} \subseteq X$ [159]. Résoudre un CSP $\langle X, D, C \rangle$, c'est trouver, si elles existent, les instanciations de X ou, autrement dit, les n-uplets de valeurs $\langle v_1, \ldots, v_n \rangle$ avec $v_i \in d_i = dom(x_i)$, $x_i \in X$, $d_i \in D$, qui satisfont toutes les contraintes $c_j \in C$. Cette tâche est NP complète. Pour une description détaillée du domaine de la satisfaction des contraintes, le lecteur peut consulter [70, 74, 20].

Propagation des contraintes

Une méthode pour établir la satisfiabilité d'un problème de satisfaction de contraintes, quand l'espace de recherche (D) est fini, consiste en la recherche systématique d'une solution [41]. Pour réduire l'espace de recherche et le coût général de recherche, on utilise des algorithmes de propagation des contraintes. Le principe est de remplacer un *CSP* donné par un autre, équivalent par rapport à l'ensemble de solutions, mais plus facile à résoudre [41]. Dans cette optique, le domaine D est systématiquement réduit, en éliminant les valeurs qui ne participent pas à au moins une solution du système. Pour cela, la propriété d'arc-consistance généralisée (GAC) (voir définition 6.1) est imposée à chaque contrainte.

Définition 6.1. ***(GAC)*** *Une contrainte $C(X)$ est généralement arc-consistante si pour tout $x_k \in X$ et tout $a \in D_k$*

$$\exists d \in C(X) \text{ tel que } d[x_k] = a.$$

En d'autres mots, chaque valeur du domaine doit participer à une solution locale.

Imposer l'arc consistance généralisée garantit la satisfaction des contraintes *localement.* Pour trouver une solution du *CSP* il faut trouver une instanciation totale (*i.e.* une valeur pour chaque variable de X) telle que *toutes* les contraintes de C soient satisfaites. A ce jour, deux types d'algorithmes de résolution existent : rétrospectifs et prospectifs. Les algorithmes rétrospectifs, tels que backtrack, backjump, backchecking, backmarking, dynamic backtracking, *etc.*, vise une instanciation des variables, construite de façon incrémentalement. Lorsqu'il n'y a plus de valeurs

[7] Constraint Satisfaction Problem

possible pour une variable en cours d'instanciation, on revient en arrière. Les algorithmes prospectifs, comme forward-checking, partial look-ahead, full look-ahead *etc.*, à chaque étape $k, k \in 1..n$ cherchent à instancier la variable x_k, ensuite filtrent les domaines des variables x_i, $i > k$ et si ils trouvent une variable x_i pour laquelle le domaine est vide, essayent la valeur suivante pour x_k s'il y en a une, sinon suit à un retour en arrière on reprend le processus pour les valeurs suivantes de $x_{k-1}, x_{k-2} \ldots$

6.4.4.1 Modélisation des relations topologiques comme les variables d'un *CSP*

L'approche conventionnelle de raisonnement spatial qualitatif à base de réseaux de contraintes considère les régions de l'espace comme des variables dont les domaines ($\mathbb{R}^2$ pour l'espace 2D) sont infinis. En revanche, Brand considère les régions de l'espace topologique comme des constantes, regroupées dans l'ensemble *Regions* [41]. La relation topologique entre deux régions est considérée comme une variable ayant comme domaine un sous-ensemble de $\mathcal{T}$ (l'ensemble $\mathcal{T}$ est défini dans la section 6.4.2). Une telle variable existe pour tout paire ordonnée de régions :

$$\text{variable } Topology[A,B] \text{ où } dom(Topology[A,B]) \subseteq \mathcal{T} \text{ et } A,B \in Regions.$$

Les contraintes imposées sur les variables $Topology[A,B]$ représentent les propriétés inter-relations définies par le tableau de composition des relations topologiques (voir la figure 2.6, page 24) et par le tableau des relations inverses (voir le tableau 6.9). Supposons que la fonction *convTop* renvoie pour chaque relation topologique son inverse, alors pour chaque paire de régions (A,B) on définit la contrainte :

$$convT_{(A,B)} : Topology[A,B] \rightarrow Topology[B,A] \in convTop(Topology[A,B]), \forall \{A,B\} \subseteq Regions. \tag{6.4}$$

De façon analogue, si la fonction *compTop* renvoie, pour une paire de relations topologiques R_1 et R_2, les relations qui peuvent être obtenues en composant R_1 et R_2, alors pour chaque paire de régions (A,B) on définit la contrainte :

$$\begin{aligned} compT_{(A,B)} \quad : \quad & Topology[A,B] \wedge Topology[B,C] \rightarrow \\ & Topology[A,C] \in compTop\,(Topology[A,B], Topology[B,C]) \rightarrow \\ & \forall \{A,B,C\} \subseteq Regions \end{aligned} \tag{6.5}$$

Ces contraintes représentent le *modèle de contraintes topologiques* qui décrit la structure d'une classe de problèmes. Elles ne changent pas d'une exécution à l'autre.

La deuxième classe de contraintes, qu'on nommera *données topologiques*, spécifie un problème particulier qui peut changer d'une exécution à une autre. Les *données topologiques* sont dérivées à partir de la base de connaissances *BC*. Concrètement, pour chaque paire d'objets spatiaux (A,B), si un ou plusieurs tuples $topRel\,(A,B)$, $topRel \in \mathcal{T}$, sont explicitement définis dans la base *BC*, alors le domaine de $Topology[A,B]$ est initialisé à $\{topRel | topRel \in \mathcal{T} \wedge \langle A,B \rangle \in (topRel)^{OP}\}$. Dans le cas contraire, le domaine de $Topology[A,B]$ est initialisé à $\mathcal{T}$. Ainsi, le réseau de contraintes contient n^2 variables, où n est le nombre d'objets spatiaux définis dans la base de connaissances exploitée, *BC*. Par exemple, si la base *BC* contient les tuples `overlaps`(C,D) et `touches`(C,D) alors le domaine de la variable $Topology[C,D]$ est défini comme étant {`overlaps`, `touches`}.

De plus, le domaine des variables $Topology[A,A]$ est restreint à $\{\texttt{equals}\} : Topology[A,A] = \{\texttt{equals}\}, \forall A \in Regions$.

En modélisant les relations topologiques comme des variables et les informations statiques (les régions) comme des constantes, l'approche relation-variable de Brand équivaut à un CSP classique dont le domaine est fini. Elle est indépendante du *solveur* employé et de la méthode de résolution de contraintes utilisée [41]. En termes de complexité de calcul, si on traite un ensemble de n régions, la complexité de l'algorithme de vérification de l'arc consistance généralisée est $O(n^3)$, si on considère que GAC est imposé en temps constant pour chaque contrainte $compTop(Topology[a,b], Topology[b,c], Topology[a,c])$ [41].

Par défaut, le système vérifie exclusivement l'arc consistance généralisée du *CSP* car cela se fait dans un temps polynomial. Néanmoins, le raisonnement peut être poussé encore plus loin, en laissant à l'utilisateur le choix de commander la résolution du *CSP* à la place de la vérification de l'arc consistance généralisé, en sachant que la résolution d'un *CSP* se fait en temps exponentiel dans le nombre des variables, dans notre cas n^2 (n est le nombre de régions décrites dans la base *BC*). La résolution a deux avantages principaux :

- si aucune solution n'est découverte, alors on peut déduire que la base *BC* contient des informations incohérentes, ou contradictoires.
- si une ou plusieurs solutions $S_i = \langle Topology_i[A_1,A_1], Topology_i[A_1,A_2], \ldots, Topology_i[A_n,A_n]\rangle$, $i \geq 1$ existent, alors pour toute paire de régions (A,B), la relation topologique entre A et B est construite comme disjonction $\cup_{i\geq 1} Topology_i[A,B]$. Ainsi, dans certains cas, il est possible de réduire davantage les domaines des variables topologiques.

6.4.4.2 Modélisation des relations de direction comme les variables d'un *CSP*

Les relations de direction entre deux régions A et $B \in Regions$ sont modélisées comme des variables ayant comme domaine l'ensemble $\mathcal{D} = \delta(North, West, South, East, Neutral)$. Une telle variable existe pour toute paire de régions :

$$\text{variable } Direction[A,B] \text{ où } Direction[A,B] \subseteq \delta(North, West, South, East, Neutral) \text{ et } A,B \in Regions.$$

Les contraintes imposées sur les variables $Direction[A,B]$ représentent les propriétés inter-relations, définies par les tableaux de composition des relations de direction, qui peuvent être construites en utilisant les algorithmes proposées dans [224], et par les tableaux de relations inverses qui peuvent être construites en utilisant le théorème 2.3. Ainsi, en supposant que la fonction *convDir* renvoie l'ensemble de relations inverses qui correspondent à la relation de direction qu'elle reçoit comme paramètre, pour chaque paire de régions (A,B), on définit la contrainte :

$$convD_{(A,B)} : Direction[A,B] \rightarrow Direction[B,A] \in convDir(Direction[A,B]), \forall \{A,B\} \subseteq Regions. \tag{6.6}$$

De la même façon, si le tableau de composition est représenté par la fonction binaire *compDir*, alors, pour chaque paire de régions (A,B), on définit la contrainte :

$$compD_{(A,B)} : Direction[A,B] \wedge Direction[B,C] \rightarrow Direction[A,C] \in compDir(Direction[A,B], Direction[B,C]), \forall \{A,B,C\} \subseteq Regions. \quad (6.7)$$

Par exemple, pour des régions de type REG^*, le tableau 6.10 définit l'inverse de chaque relation de direction de type *carreau unique*. Celui-ci a été construit en utilisant le lemme 2.1 (page 35). Pour le même type de région, le tableau 6.11, illustre le résultat de la composition des relations de direction de type *carreau unique*. Celui-ci a été construit en utilisant les algorithmes proposés dans [224]. Les tableaux de composition et de relations inverses peuvent être adaptés pour des géométries moins complexes.

Relation	Inverse
N	$\{S, W:S, W:E, S:E, W:S:E\}$
W	$\{E, N:S, N:E, S:E, N:S:E\}$
S	$\{N, W:E, N:W, N:E, N:W:E\}$
E	$\{W, N:S, N:W, W:S, N:W:S\}$
B	$\{N:W, N:S, N:E, N:B, W:S, W:E, W:B, S:E, S:B, E:B, N:W:S, N:W:E, N:W:B, N:S:E, N:S:B, N:E:B, W:S:E, W:S:B, W:E:B, S:E:B, N:W:S:E, N:W:S:B, N:S:E:B, W:S:E:B, N:W:S:E:B\}$

TAB. *6.10: Tableau d'inverses pour les relations de type carreau unique.*

$\circ$	N	W	S	E	B
N	N	$\delta(N,W,E,B)$	U_{dir}	$\delta(N,W,E,B)$	$\delta(N,W,E,B)$
S	U_{dir}	$\delta(S,W,E,B)$	S	$\delta(S,W,E,B)$	$\delta(S,W,E,B)$
W	$\delta(W,N,S,B)$	W	$\delta(W,S,N,B)$	U_{dir}	$\delta(W,S,N,B)$
E	$\delta(E,N,S,B)$	U_{dir}	$\delta(E,N,S,B)$	E	$\delta(E,N,S,B)$
B	$\delta(N,W,E)$	$\delta(W,S,N)$	$\delta(S,W,E)$	$\delta(E,N,S)$	B

TAB. *6.11: Tableau de composition pour les relations de type carreau unique.*

Pour les paires d'objets spatiaux (A,B) dont la relation de direction $Q \in 2^{\mathcal{D}}$ est explicitement contenue dans la base de connaissances, le domaine de la variable $Direction[A,B]$ est réduit à $\{R_i | Q = \cup_{i=1}^{k} R_i\}$. Une fois le réseau de contraintes construit, en utilisant un *solveur* qui impose l'arc consistance généralisée (GAC), on réduit autant que possible les domaines des variables *Direction*. Le réseau de contraintes ainsi raffiné sera utilisé pour mettre à jour la base de connaissances *BC*.

Comme dans le cas des relations topologiques, les domaines des variables $Direction[A,B]$, $A,B \in Regions$ peuvent être réduits davantage en imposant la résolution du *CSP* à la place de la vérification de l'arc consistance généralisée.

6.4.4.3 Modélisation des relations de distance comme les variables d'un *CSP*

Les relations de distance entre deux objets A et B peuvent également être modélisées comme des variables d'un *CSP*. Ainsi, pour toute paire d'objets spatiaux (A,B), on définit une variable représentant la relation de distance entre A et B comme suit :

$$\text{variable } Distance\,[A,B] \text{ où } Distance\,[A,B] \in \Lambda \text{ et } A,B \in Regions.$$

Le *modèle de contraintes de distance* est constitué à partir des propriétés inter-relations définies par les tableaux de composition et à partir du tableau de relations inverses des relations de distance. Comme le tableau de composition des relations de distance est défini en fonction des relations de direction, nous définissons les fonctions $compDist_1$ et $compDist_2$[8] qui renvoient l'ensemble de relations de distance qui peuvent exister entre deux régions A et B en fonction des relations de distance que A vérifie par rapport à une troisième région C, que C vérifie par rapport à B, et aussi en fonction de la direction entre A et C et A et B :

$$\begin{aligned} compDist1_{(A,C,B)} \;:\; & Distance\,[A,C] \wedge Distance\,[C,B] \wedge Direction\,[A,C] = Direction\,[C,B] \\ & \rightarrow Distance[A,B] \in compDist_1(Distance\,[A,C]\,, Distance\,[C,B]), \\ & \forall A,B,C \in Regions \end{aligned} \tag{6.8}$$

$$\begin{aligned} compDist2_{(A,C,B)} \;:\; & Distance\,[A,C] \wedge Distance\,[C,B] \wedge Direction\,[A,C] = -Direction\,[C,B] \\ & \rightarrow Distance[A,B] \in compDist_2(Distance\,[A,C]\,, Distance\,[C,B]), \\ & \forall A,B,C \in Regions. \end{aligned} \tag{6.9}$$

En ce qui concerne le tableau des relations inverses, il est construit différemment en fonction des propriétés de la distance considérée (*i.e.* en fonction des propriétés de symétrie de la distance employée). Nous avons décidé de le définir au niveau des *données de distance* et non pas au niveau du *modèle de contraintes de distance*. Ainsi, le raisonnement peut être adapté au type de distance pris en compte. De plus, au niveau des *données de distance*, nous prenons en compte exclusivement les relations de distance décrites dans la base de connaissances *BC* qui sont rapportées à la même échelle que *S*. Ces relations de distance sont utilisées pour restreindre le domaine des variables *Distance* qui leur correspond.

6.4.4.4 Raisonnement spatial qualitatif à base de relations topologiques, de direction et de distance

Comme nous l'avons illustré dans la section précédente, pour inférer des relations spatiales implicites à partir de relations spatiales qualitatives connues et de règles de composition, nous avons adopté l'approche proposée par Brand [41, 42, 22]. Celle-ci s'appuie sur la modélisation du domaine spatial à travers un réseaux de contraintes, organisé selon deux niveaux : les connaissances

[8] Nous rappelons que $\forall D = R_1 : \ldots : R_k$, où $R_i \in \{North, West, South, East, Neutral\}$, $\forall i \in 1..k : -D = R_1^{\downarrow} : \ldots : R_k^{\downarrow}$. Pour plus de détails voir section 2.3.2.4.

axiomatiques, qui ne changent pas d'un problème à un autre (que nous avons appelé *modèle de contraintes*), et les connaissances spécifiques à la configuration spatiale courante (que nous avons appelé *données*).

L'un des avantages majeurs de cette approche de gestion des relations spatiales à travers des variables de *CSP*, est la facilité offerte pour ajouter de nouvelles contraintes, et surtout la possibilité de combiner les différents types de raisonnement (à base de topologie, de direction et de distance).

Par exemple, pour combiner le raisonnement topologique avec le raisonnement à base de direction, il suffit d'ajouter dans la partie du *modèle de contraintes* des règles telles que :

$Topology\,[A,B] \in \{\texttt{equals}, \texttt{contains}, \texttt{tangent\_contains}\} \rightarrow Direction\,[B,A] = \{\texttt{Neutral}\}$

$Topology\,[A,B] \in \{\texttt{inside}, \texttt{tangent\_inside}\} \rightarrow Direction\,[B,A] \subseteq \theta$, où

θ = { `Neutral`, `North:Neutral`, `West:Neutral`, `South:Neutral`, `East:Neutral`, `North:West:Neutral`, `North:South:Neutral`, `North:East:Neutral`, `North:West:South:Neutral`, `North:West:East:Neutral`, `West:South:East:Neutral`, `North:West:South:East:Neutal` }.

La première règle dit que si une région *A* est *égale à* ou *inclut* la région *B*, alors *B* est incluse dans le rectangle englobant minimal de *A*, donc la relation de direction de *B* par rapport à *A* est la relation neutre. Si la région *A* est *incluse* dans la région *B*, alors une partie de *B* appartient au rectangle englobant minimal de *A*, donc la relation de direction de *B* par rapport à *A* doit passer par le carreau `Neutral`.

On peut également considérer des règles qui combinent le raisonnement à base de direction avec le raisonnement topologique, telles que :

$Direction[A,B] \in 2^{\delta(\texttt{North},\texttt{West},\texttt{South},\texttt{East})} \rightarrow Topology[A,B] \in \{\texttt{disjoint}, \texttt{touches}\}$.

En d'autres mots, si la relation de direction entre deux régions *A* et *B* ne contient pas le carreau `Neutral`, alors les deux régions sont disjointes.

Un autre exemple de contrainte qui combine le raisonnement topologique avec le raisonnement à base de distance est :

$Topology\,[A,B] \in \{\texttt{equals}, \texttt{contains}, \texttt{tangent\_contains}, \texttt{inside}, \texttt{tangent\_inside}\}$

$\rightarrow Distance\,[A,B] \in \{\texttt{veryClose}\}$.

Cette règle exprime le fait que, si une région *A* est *égale à, inclut*, ou *est incluse dans* une autre région *B*, on peut considérer les deux régions comme très proches.

Il est également possible de définir des règles qui combinent les trois types de relations. Par exemple, pour déclarer que si deux objets *B* et *C* touchent l'objet *A* et ont, en plus, la même orientation par rapport à celui-ci, alors *B* et *C* sont très proches l'un de l'autre, on utilise la règle :

$Topology\,[A,B] = \{\texttt{touches}\} \wedge Topology\,[A,C] = \{\texttt{touches}\} \wedge Direction\,[A,B] = Direction\,[A,C]$

$\rightarrow Distance\,[B,C] = \{\texttt{veryClose}\}$

Cependant, cette règle n'est pas toujours vraie, étant dépendante de la dimension des objets comparés, du type de distance pris en compte, ainsi que de l'échelle utilisée. Il est possible de définir une telle règle dans un problème particulier. Ce genre de lien entre les relations spatiales peut être ajouté dans la partie *données* du *CSP* afin de faciliter la déduction de relations spatiales dans des cas spécifiques qu'ils caractérisent.

6.4.5 Ontologie de relations spatiales de référence pour ONTOAST

Comme nous l'avons montré dans le chapitre 3.3, il n'existe pas à ce jour d'ontologie de relations spatiales qui soit proposée comme ontologie de référence ou comme recommandation pour la description des relations spatiales dans le cadre du Web Sémantique. Dans ce sens, afin de garantir la cohérence de la traduction des descriptions spatiales de OWL vers ONTOAST

et de ONTOAST vers OWL, nous avons défini une ontologie de relations spatiales qualitatives, `QualitativeSpatialRelations`, que nous prenons comme référence. Ainsi, lors de l'importation d'une ontologie OWL, O_1, l'analyseur de ONTOAST peut construire des instances de relations spatiales (topologiques, de direction et de distance) mais seulement, à ce stade de nos travaux, pour les descriptions réalisées par rapport à l'ontologie `QualitativeSpatialRelations`. Si l'ontologie O_1 contient des descriptions spatiales réalisées par rapport à une autre ontologie, par exemple l'ontologie `SpatialRelations` (voir section 3.3.2), celles-ci ne sont pas traduites de façon adaptée, comme des tuples attachés aux relations spatiales prédéfinis de ONTOAST, mais comme des tuples de nouvelles associations. En conséquence, ces informations ne sont pas prises en compte pour l'inférence de nouvelles relations. Une perspective intéressante dans ce sens est d'étudier *i)* la définition des alignements entre différentes ontologies décrivant des relations spatiales qualitatives, mais aussi *ii)* la spécification d'un ensemble de règles de traduction des données décrites par rapport à une ontologie de relations spatiale O_1 vers des données équivalentes mais définies par rapport à une autre ontologie O_2.

L'exportation des relations spatiales contenues dans des bases de connaissances ONTOAST se fait par rapport à la même ontologie, `QualitativeSpatialRelations`. Celle-ci est illustrée dans la figure 6.18.

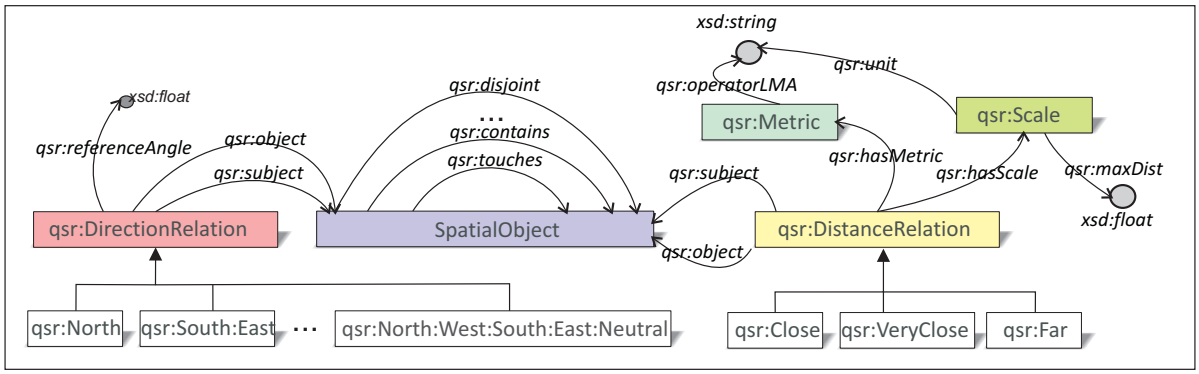


FIG. *6.18: L'ontologie de relations spatiales utilisée par ONTOAST.*

Nous avons modélisé les *relations topologiques* comme des *propriétés-lien*, dont le *domaine* et l'*image* désignent des objets spatiaux, instances de `SpatialObject`. La propriété `spatiallyRelated` est l'équivalent de la relation topologique homonyme définie dans le modèle spatial de ONTOAST. Elle dispose de huit sous-propriétés qui correspondent aux relations du calcul RCC8 : `contains`, `overlap`, `inside`, `tangent_contains`, `tangent_inside`, `equals`, `disjoint`, `touches`. A l'aide d'axiomes de propriétés, nous avons défini, pour chaque relation topologique, son inverse (voir annexe C).

Les *relations de direction* ainsi que les *relations de distance*, modélisées en ONTOAST à travers des associations binaires et quaternaires, sont traduites en OWL à l'aide de la réification. La raison est que le langage OWL traite exclusivement des relations binaires, modélisées comme des *propriétés*. Ainsi, la classe `DirectionRelation` représente une association binaire entre deux objets spatiaux et une relation de direction. Les objets spatiaux sont reliées à la classe `DirectionRelation` à travers les propriétés-lien `subject` et `object`, tandis que l'angle de référence attaché aux relations de direction est spécifié par l'attribut `referenceAngle`. Les 31 rela-

tions de direction prises en compte par l'approche *CDR* sont définies comme des specialisations de la classe `DirectionRelation` (voir figure 6.18).

La classe `DistanceRelation` modélise les relations de distance. Elle dispose de trois spécialisations : `VeryClose`, `Close` et `Far`. Une relation de distance est définie par rapport à une métrique, instance de la classe `Metric`, et par rapport à une échelle, instance de la classe `Scale` (voir figure 6.18). Pour une métrique donnée, on peut spécifier, en utilisant l'*attribut* `operatorLMA`, l'opérateur[9] LMA qu'il faut utiliser pour calculer la distance entre deux régions *A* et *B* à partir de leurs descriptions géométriques. Pour une échelle donnée, on peut préciser l'unité de mesure utilisée, à l'aide de l'attribut `unit`, ainsi que la distance maximale prise en compte, à l'aide de l'attribut `maxDist`.

6.4.6 Étude de cas

Dans cette section nous présentons un exemple d'utilisation du système ONTOAST. Pour cela, nous considérons l'ontologie `qsrEx` illustrée dans la figure 6.19. Celle-ci contient des descriptions spatiales qualitatives réalisées par rapport à l'ontologie `QualitativeSpatialRelations`, et des descriptions spatiales quantitatives, définies par rapport à l'ontologie `GeoRSSSimple`. Concrètement, l'ontologie `qsrEx` décrit la configuration spatiale illustrée dans la figure 6.21. Pour le département : `D1` ainsi que pour le lac `L`, on dispose des données géométriques. Les relations topologiques `contains(D2, C)` et `touches(D1, D2)` sont explicitement décrites par l'ontologie `qsrEx`, ainsi que les relations de direction `South(D3, D1)` et `South(D1, D2)`.

```
…
Declaration(Class(qsrEx:Department))
SubClassOf(qsrEx:Department qsr:SpatialObject)

Declaration(DataProperty(qsrEx:surface))
DataPropertyDomain(qsrEx:surface qsrEx:Department)
DataPropertyRange(qsrEx:surface xsd:float)

Declaration(DataProperty(qsrEx:population))
DataPropertyDomain(qsrEx:population qsrEx:Department)
DataPropertyRange(qsrEx:population xsd:integer)

Declaration(DataProperty(qsrEx:hasName))
DataPropertyDomain(qsrEx:hasName qsrEx:Lake)
DataPropertyDomain(qsrEx:hasName qsrEx:Department)
DataPropertyDomain(qsrEx:hasName qsrEx:City)
DataPropertyRange(qsrEx:hasName xsd:string)

Declaration(Class(qsrEx:Lake))
SubClassOf(qsrEx:Lake qsr:SpatialObject)

Declaration(Class(qsrEx:City))
SubClassOf(qsrEX:City qsr:SpatialObject)

Declaration(Individual(qsrEx:C))
ClassAssertion(qsrEx:C qsrEx:City)

Declaration(Individual(qsrEx:D1))
ClassAssertion(qsrEx:D1 qsrEx:Department)

ObjectPropertyAssertion(qsr:touches qsrEx:D1 qsrEx:D2)
ObjectPropertyAssertion(qsr:object qsrEx:D1 qsrEx:south1)
ObjectPropertyAssertion(qsr:subject qsrEx:D1 qsrEx:south2)
DataPropertyAssertion(georss:polygon qsrEx:D1
                      "45.716 6.487
                       45.691 6.454
                       45.6549 6.4568
                       ...
                       45.716 6.487
                       "^^xsd:string)

Declaration(Individual(qsrEx:L))
ClassAssertion(qsrEx:L qsrEx:Lake)
DataPropertyAssertion(georss:polygon qsrEx:L
                      "45.8017  5.8274
                       45.7967  5.8166
                       45.7618  5.8317
                       ...
                       45.8017  5.8274")

Declaration(Individual(qsrEx:D2))
ClassAssertion(qsrEx:D2 qsrEx:Department)
ObjectPropertyAssertion(qsr:contains qsrEx:D2 qsrEx:C)
ObjectPropertyAssertion(qsr:object qsrEx:D2 qsrEx:south2)

Declaration(Individual(qsrEx:D3))
ClassAssertion(qsrEx:D3 qsrEx:Department)
ObjectPropertyAssertion(qsr:subject qsrEx:D3 qsrEx:south1)
…
```

FIG. *6.19: Exemple d'ontologie contenant des descriptions spatiales réalisées à l'aide des ontologies* `GeoRSSSimple` *(préfixe* `geoRSS`*) de* `QualitativeSpatialRelations` *(préfixe* `qsr`*). Pour cet exemple, nous avons utilisé la syntaxe fonctionnelle de OWL 2 pour des raisons de concision des descriptions.*

[9] Le terme *opérateur* est utilisé ici pour désigner un programme qui calcule différents types de distances entre régions, et non pas dans le sens d'opération mathématique.

```
…
class: Department
super-class: SpatialObject
variables:
  variable: surface
    type: float
  variable: population
    type: integer
  variable: hasName
    type: string

class: Lake
super-class: SpatialObject
variables:
  variable: hasName
    type: string

class: City
super-class: SpatialObject
variables:
  variable: hasName
    type: string
```

```
/* Instances section */
instance: D1
is-a: Department
  Geom = "#45.716 6.487,
          45.691 6.454,
           ...
          45.716 6.487#"
instance: L
is-a: Lake
  Geom = "# 45.8017  5.8274
          45.7967  5.8166
          45.7618  5.8317
          ...
          45.8017  5.8274#"
instance: C
is-a: City

instance: D3
is-a: Department

instance: D2
is-a: Department
```

```
/* Tuples section */
tuple:
 is-a:contains
   subject = D2
   object = C

tuple:
 is-a:South
   subject = D3
   object = D1

tuple:
 is-a:South
   subject = D1
   object = D2

tuple:
 is-a:touches
   subject=D1
   object= D2
…
```

FIG. *6.20: Résultat de la traduction de l'ontologie* `qsrEx.owl` *en ONTOAST.*

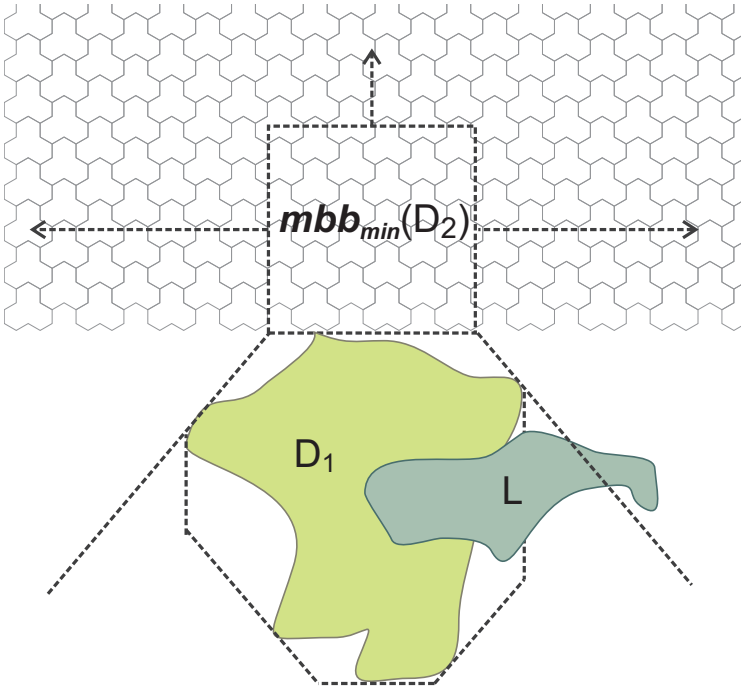


FIG. *6.21: Configuration spatiale décrite par l'ontologie* `qsrEx`*. La géométrie du département D2 n'est pas connue, mais les limites inférieures de* `mbb(D2)` *peuvent être déduites à partir de l'octogone englobant minimal de* `D1`*. La figure 6.19 mentionne une ville,* `C`*, et un troisième département,* `D3`*, qui ne sont pas non plus dessinés ici, car on ne dispose pas de leurs géométries.*

À partir des informations contenues par l'ontologie `qsrEx.owl`, on veut répondre aux requêtes :

a) quelle est la relation topologique entre la ville `C` et le département `D2` ?

b) quelle est la relation topologique entre le département `D1` et le lac `L` ?

c) quelle est la relation de direction entre le département `D3` et le département `D2` ?

d) quelle est la relation topologique entre le département `D3` et le département `D1` ?

Afin de pouvoir traiter ces requêtes, il faut, dans un premier temps, traduire l'ontologie `qsrEx` en ONTOAST, à l'aide du traducteur XSLT présenté dans la section 5.5 et adapté pour gérer de façon dédiée les informations décrites à l'aide des ontologies *GeoRSS-Simple*, *OWL-Time* et *QualitativeSpatialRelations*. Le résultat de cette traduction est illustré dans la figure 6.20.

Ensuite, en utilisant l'algorithme de déduction d'une relation topologique `SpatialRelation` (voir page 173), on déduit la relation `inside` entre la vile `C` et le département `D2` à partir de la relation `contains(D2, C)` contenue par l'ontologie `qsrEx` et en utilisant le tableau des relations inverses défini pour les relations topologiques (tableau 6.9, page 172).

Lors du traitement de la deuxième requête (*b)*), la relation topologique entre le lac `L` et le département `D1` est calculée à partir des données géométriques (voir section 6.4.3.1). La relation résultat, `overlap(L,D1)` est ajoutée à l'ontologie `qsrEx`.

Pour la déduction de la relation de direction entre les départements `D3` et `D2` (requête *c)*), l'algorithme `SpatialRelation` utilise les relations explicites `South(D1, D2)` et `South(D3, D1)` ainsi que le tableau de composition des relations de direction (tableau 6.11, page 182). La relation résultat - `South(D3, D2)` - est ajoutée à l'ontologie `qsrEx`.

Étant donné le fait qu'on ne peut pas déduire la relation topologique entre `D3` et `D1` à partir de son inverse, ou en utilisant le tableau de composition des relations topologiques (aucune relation topologique qui implique `D3` n'est disponible), ni à partir des calculs géométriques (on ne dispose pas de la géométrie de `D3`), l'algorithme `SpatialRelation` construit le réseau de contraintes *CtNetwork*. Celui-ci contient, au niveau *données*, les variables topologiques, de direction et de distance, dont les domaines initiaux sont définis par les tableaux 6.12, 6.13 et 6.14. Les ensembles $\mathcal{T}$, $\mathcal{D}ir$ et Λ sont définis dans la section 6.4.2 (voir page 171).

Topology	D1	D2	D3	C	L
D1	equals	touches	$\mathcal{T}$	$\mathcal{T}$	$\mathcal{T}$
D2	$\mathcal{T}$	equals	$\mathcal{T}$	contains	$\mathcal{T}$
D3	$\mathcal{T}$	$\mathcal{T}$	equals	$\mathcal{T}$	$\mathcal{T}$
C	$\mathcal{T}$	$\mathcal{T}$	$\mathcal{T}$	equals	$\mathcal{T}$
L	overlaps	$\mathcal{T}$	$\mathcal{T}$	$\mathcal{T}$	equals

TAB. *6.12: Les domaines initiaux des variables topologiques, construits à partir de l'ontologie* `qsrEx`.

Direction	D1	D2	D3	C	L
D1	Neutral	South	$\mathcal{D}ir$	$\mathcal{D}ir$	$\mathcal{D}ir$
D2	$\mathcal{D}ir$	Neutral	$\mathcal{D}ir$	$\mathcal{D}ir$	$\mathcal{D}ir$
D3	South	$\mathcal{D}ir$	Neutral	$\mathcal{D}ir$	$\mathcal{D}ir$
C	$\mathcal{D}ir$	$\mathcal{D}ir$	$\mathcal{D}ir$	Neutral	$\mathcal{D}ir$
L	$\mathcal{D}ir$	$\mathcal{D}ir$	$\mathcal{D}ir$	$\mathcal{D}ir$	Neutral

TAB. *6.13: Les domaines initiaux des variables de direction, construits à partir de l'ontologie* `qsrEx`.

Distance	D1	D2	D3	C	L
D1	veryClose	Λ	Λ	Λ	Λ
D2	Λ	veryClose	Λ	Λ	Λ
D3	Λ	Λ	veryClose	Λ	Λ
C	Λ	Λ	Λ	veryClose	Λ
L	Λ	Λ	Λ	Λ	veryClose

TAB. *6.14: Les domaines initiaux des variables de distance, construits à partir de l'ontologie* `qsrEx`.

En imposant l'arc consistance pour le réseau de contraintes *CtNetwork*, on tente de réduire les domaines des variables. Ainsi, à partir des contraintes (6.4) (page 180), qui utilisent le tableau des relations topologiques inverses 6.9, on peut réduire les domaines des variables $Topology[C,D2]$, $Topology[D2,D1]$, $Topology[D1,L]$ comme suit :

$Topology[D2,C] = contains \rightarrow Topology[C,D2] \in convTop(contains) = \{inside\}$

$Topology[D1,D2] = touches \rightarrow Topology[D2,D1] \in convTop(touches) = \{touches\}$

$Topology[L,D1] = overlaps \rightarrow Topology[D1,L] \in convTop(overlaps) = \{overlaps\}$

De façon analogue, à l'aide des contraintes (6.6) (page 181) qui utilisent le tableau des inverses des relations de direction, on peut réduire les domaines des variables $Direction[D2,D1]$ et $Direction[D1,D3]$ comme suit :

$Direction[D1,D2] = South \rightarrow Direction[D2,D1] \in convDir(South) = \{North, West:East,$
$North:West, North:East, North:West:East\}$

$Direction[D3,D1] = South \rightarrow Direction[D1,D3] \in convDir(South) = \{North, West:East,$
$North:West, North:East, North:West:East\}$

En utilisant un raisonnement analogue, à l'aide des contraintes de composition définies par les équations (6.5) (page 180) et (6.7) (page 182) on obtient :

$Topology[D1,D2] = touches \wedge Topology[D2,C] = contains \rightarrow$
$Topology[D1,C] \in compTop(touches, contains) = \{disjoint\}$

$Topology[L,D1] = overlap \wedge Topology[D1,D2] = touches \rightarrow$
$Topology[L,D2] \in compTop(overlap, touches) = \{disjoint, touches,$
$overlap, tangent\_contains, contains\}$

$Direction[D3,D1] = South \wedge Direction[D1,D2] = South \rightarrow$
$Direction[D2,D2] \in compDir(South, South) = \{South\}$

En utilisant des contraintes qui mélangent relations topologique et relations de direction, comme :
$Direction[A,B] \in 2^{\delta(\texttt{North},\texttt{West},\texttt{South},\texttt{East})} \rightarrow Topology[A,B] \in \{\texttt{disjoint},\texttt{touches}\}$,
$\forall A,B \in Regions$,
on peut réduire le domaine de la variable `Topology[D3,D1]` :

$Direction[D3,D1] = South \rightarrow Topology[D3,D1] \in \{disjoint, touches\}$.

Les domaines des variables de distance ne peuvent pas être réduites en utilisant des contraintes de composition, car aucune relation de distance n'est explicite. Néanmoins, la relation de distance entre les régions `C` et `D2` peut être déduite à partir de la contrainte :

$Topology[C,D1] = inside \rightarrow Distance[C,D1] \in \{veryClose\}$

qui mélange relations topologiques et relations de distance (voir section 6.4.4.4, page 183).

Pour les nouveaux domaines, le processus de réduction recommence jusqu'à ce que les domaines ne changent plus. Les domaines réduits qu'on obtient suite à ce processus sont donnés dans les tableaux 6.15, 6.16 et 6.17.

Topology	D1	D2	D3	C	L
D1	equals	touches	disjoint, touches	disjoint	overlap
D2	touches	equals	$\mathcal{T}$	contains	disjoint, touches, overlap inside, tangent_inside
D3	disjoint, touches	$\mathcal{T}$	equals	$\mathcal{T}$	$\mathcal{T}$
C	disjoint	inside	$\mathcal{T}$	equals	$\mathcal{T}$
L	overlap	disjoint, touches, overlap tangent_contains, contains	$\mathcal{T}$ $\mathcal{T}$	$\mathcal{T}$	equals

TAB. *6.15: Les domaines réduits des variables topologiques.*

Direction	D1	D2	D3	C	L
D1	Neutral	South	North, West :East, North :West, North :East, North :West :East	$\mathcal{D}ir$	$\mathcal{D}ir$
D2	North, West :East, North :West, North :East, North :West :East	Neutral	$\mathcal{D}ir$	$\mathcal{D}ir$	$\mathcal{D}ir$
D3	South	$\mathcal{D}ir$	Neutral	$\mathcal{D}ir$	$\mathcal{D}ir$
C	$\mathcal{D}ir$	$\mathcal{D}ir$	$\mathcal{D}ir$	Neutral	$\mathcal{D}ir$
L	$\mathcal{D}ir$	$\mathcal{D}ir$	$\mathcal{D}ir$	$\mathcal{D}ir$	Neutral

TAB. *6.16: Les domaines réduits des variables de direction.*

Distance	D1	D2	D3	C	L
D1	veryClose	Λ	Λ	veryClose	Λ
D2	Λ	veryClose	Λ	Λ	Λ
D3	Λ	Λ	veryClose	Λ	Λ
C	veryClose	Λ	Λ	veryClose	Λ
L	Λ	Λ	Λ	Λ	veryClose

TAB. *6.17: Les domaines réduits des variables de distance, construits à partir de l'ontologie* `qsrEx`.

Pour chaque variable topologique `Topology`$[A,B], A,B \in \{D1,D2,D3,C,L\}$ dont le domaine n'est pas $\mathcal{T}$ les tuples contenus par son domaine sont ajoutés dans la base de connaissances. Par exemple, pour la variable $Topology[D3,D1]$, on ajoute dans la base de connaissances `qsrEx` les tuples `disjoint(D3, D1)` et `touches(D3, D1)`. Les mêmes mises à jour sont réalisées pour les variables `Direction`$[A,B], A,B \in \{D1,D2,D3,C,L\}$. La base de connaissances complétée avec ces nouveaux tuples peut être utilisée pour mettre à jour l'ontologie `qsrEx.owl`.

Ainsi, comme réponse pour la requête (*c)*) on obtient la disjonction des relations `disjoint` et `touches`. En d'autres termes, le département `D3` peut être relié au département `D1` par la relation topologique `disjoint` ou par la relation topologique `touches`.

6.5 Modèle temporel qualitatif en ONTOAST

De façon analogue à la description qualitative des relations spatiales, nous avons intégré dans le système ONTOAST, un modèle temporel qualitatif prédéfini, dont une représentation graphique est donnée dans la figure 6.22. Les relations temporelles en ONTOAST sont modélisées à l'aide de la méta-classe `AROMQTR` (de Arom Qualitative Temporal Relation), utilisée pour définir de façon générique une relation temporelle qualitative comme étant une association binaire dont les deux rôles (`subject` et `object`) désignent des objets temporels. Dans notre modèle, un objet temporel est une instance de la méta-classe `AROMTemporalConcept`. Comme illustré dans la figure 6.22, la méta-classe `AROMTemporalConcept` définit l'ensemble des objets ayant une dimension temporelle, modélisée par une variable dont le type est l'un des types temporels définis par AROM-ST. La méta-classe `TemporalType` est introduite exclusivement pour des raisons d'illustration, pour désigner l'ensemble de types temporels présentés dans la section 6.3.1. Il est important de préciser que l'intervalle/l'instant de temps attaché à un objet temporel peut ne pas être connu.

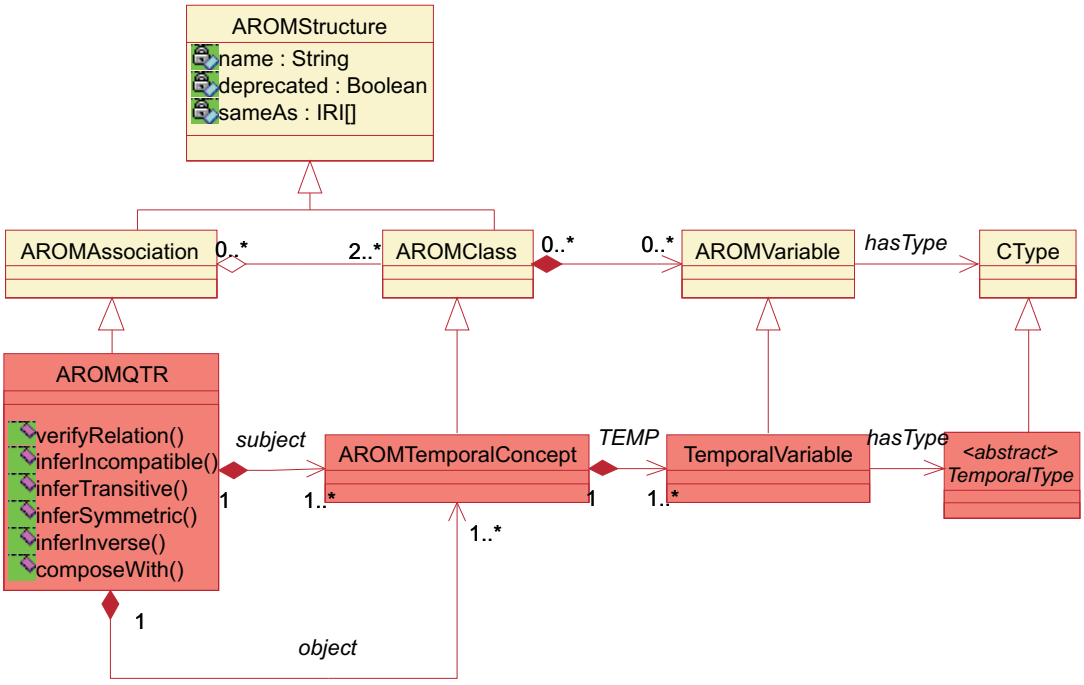


FIG. *6.22: La définition de la méta-classe* ***AROMQTR*** *qui modélise les relations temporelles qualitatives en ONTOAST.*

6.5.1 Relations temporelles en ONTOAST

Afin de modéliser en ONTOAST des relations temporelles qualitatives entre des objets temporels dont la validité est définie par des intervalles de temps, nous avons défini un modèle temporel qualitatif. Celui-ci contient les treize relations illustrées dans la figure 6.23, qui correspondent aux

relations temporelles proposées par Allen [14] (voir la description des relations de Allen en section 2.4.2).

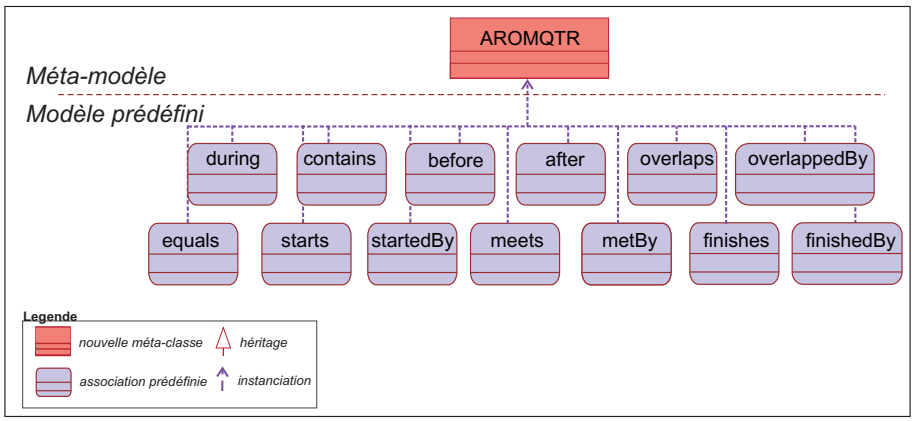


FIG. *6.23: Les relations temporelles prédéfinies ajoutées au modèle d'ONTOAST.*

Les relations temporelles sont définies pour des intervalles de temps, mais peuvent également être utilisées pour comparer des instants de temps. Néanmoins, certaines relations temporelles n'ont pas de sens lorsqu'on souhaite situer un instant par rapport à un autre, ou un instant par rapport à un intervalle. Le tableau 6.18 illustre les relations temporelles qui peuvent être définies entre deux objets dont la temporalité est exprimée à différents niveaux de complexité.

Type temporel (*objet cible*)	Type temporel (*objet de reference*)	Relations temporelles valides
`intervalle`	`intervalle`	`before`, `after`, `meets`, `metBy`, `overlaps`, `overlappedBy`, `starts`, `startedBy`, `during`, `contains`, `finishes`, `finishedBy`, `equals`
`intervalle`	`instant`	`before`, `after`, `startedBy`, `contains`, `finishedBy`
`instant`	`intervalle`	`before`, `after`, `starts`, `finishes`, `during`
`instant`	`instant`	`before`, `after`, `equals`

TAB. *6.18: Les relations temporelles qui s'appliquent entre différents types temporels.*

Pour les relations temporelles illustrées dans la figure 6.23, nous avons prédéfini la sémantique ainsi que le comportement. Cela veut dire qu'elles peuvent être instanciées sans redéfinition dans toute base de connaissances. Par exemple, pour définir une relation `during` entre deux objets temporels tels que *MyNewYorkHolidays* et *NewYorkMarathon* il suffit de définir un tuple de la relation `during` comme illustré dans la figure 6.24.

Lorsqu'une base de connaissances BC contient plusieurs tuples de relations temporelles, $R_{t_1}, \ldots, R_{t_k}$, qui relient la même paire d'objets A et B, on considère que la relation temporelle qui existe entre A et B est la disjonction $R_t = \cup_{i=1}^{k} R_{t_i}$. Par exemple, si pour la paire (*NewYorkMarathon*, *MyNewYorkHolidays*) on définit les relations `during`, `starts`, `before`, cela veut dire que *NewYorkMarathon* a eu lieu soit *pendant*, soit *au début de*, soit *avant* mon séjour à New York.

```
instance: MyNewYorkHoliday
   is-a:Event
   date=15/08/2009,14:30:00-
        17/08/2009,17:00:00
instance: NewYorkMarathon
   is-a: CharityMarathon

tuple:
   is-a: during
   subject=NewYorkMarathon
   object=MyNewYorkHoliday
```

FIG. *6.24: Définition en ONTOAST de la relation temporelle d'inclusion entre l'intervalle de temps attaché à mes vacances à NewYork et la période pendant laquelle le Marathon de New York s'est déroulé.*

6.5.2 Algorithme de déduction de relations temporelles

Afin de déduire la relation temporelle qui existe entre deux objets temporels, *A* et *B*, décrits dans une base de connaissances ONTOAST, *BC*, nous avons défini l'algorithme 3. Celui-ci admet comme paramètres d'entrée la base de connaissances de référence, *BC*, les deux objets temporels, *A* et *B*, pour lesquels on cherche la relation temporelle et l'ensemble $\mathcal{T}mp$ défini comme suit : $\mathcal{T}mp$={`before`, `after`, `meets`, `metBy`, `overlaps`, `overlappedBy`, `starts`, `startedBy`, `during`, `contains`, `finishes`, `finishedBy`, `equals`, `before`, `after`, `startedBy`, `contains`, `finishedBy`}.

L'algorithme utilise à la fois les connaissances explicites contenues dans la base *BC*, et des techniques de raisonnement temporel quantitatif et qualitatif. Il considère des objets dont la validité temporelle est représentée par des intervalles de temps, mais peut être facilement adapté pour traiter des instants.

Dans un premier temps, le système vérifie si, dans la base de connaissances *BC*, le tuple (A,B) est défini comme appartenant à l'extension d'une relation temporelle élémentaire[10] (voir lignes 1 à 3). Si c'est le cas, l'algorithme termine car la solution a été trouvée. Dans le cas où la relation entre *A* et *B* n'est pas explicite, le système cherche à la déduire à partir de sa relation inverse si celle-ci est explicite dans la base de connaissances *BC* (voir lignes 5 à 7). Concrètement, si la base de connaissance *BC* contient un tuple d'une relation temporelle, défini comme reliant les objets *B* et *A*, alors le système déduit la relation entre *A* et *B* à partir du tableau des relations inverses 2.9.

Si cela n'est pas possible, parce que la relation entre *B* et *A* n'est pas explicite ou qu'elle n'est pas élémentaire, le système vérifie si la base de connaissances *BC* contient un objet temporel *C* tel que les relations temporelles entre *A* et *C* et entre *B* et *C* soient explicites et élémentaires. Si c'est le cas, en utilisant le tableau de composition 2.10 le système déduit l'ensemble de relations temporelles possibles entre *A* et *B* (voir lignes 9 à 12). Il faut noter que le résultat de la composition peut être une relation élémentaire mais aussi une disjonction de relations élémentaires.

Si la relation temporelle entre *A* et *B* n'a pas pu être déduite à l'aide des trois premiers tests, le système teste si les intervalles de validité des objets *A* et *B* sont disponibles dans la base de connaissances *BC* (voir lignes 14 à 17). Si les données temporelles quantitatives qui décrivent *A* et *B* sont disponibles, en utilisant les opérateurs temporels du LMA, le système calcule la relation temporelle entre *A* et *B* (voir section 6.5.3).

[10]Une relation temporelle élémentaire est une relation *R* qui appartient à l'ensemble $\mathcal{T}mp$. Exemple de relation temporelle qui n'est pas élémentaire : $R = \texttt{equals} \cup \texttt{during}$.

Si, au contraire, les informations temporelles ne sont pas disponibles, le système construit, à partir des relations temporelles décrites dans la base de connaissances *BC*, un réseau de contraintes, *CtNetwork*, qui sera envoyé à un *solver*. En imposant l'arc-consistance généralisé pour le réseau de contraintes *CTNetwork*, le *solveur* affine autant que possible les relations temporelles entre les intervalles considérés. La base de connaissances sera ensuite mise à jour pour refléter les inférences réalisées. Il faut noter que, pour les paires d'intervalles pour lesquelles le *solveur* n'a pas réussi à déduire une relation temporelle autre que la conjonction de toutes les relations temporelles de $\mathcal{T}mp$, aucun tuple de relation temporelle ne sera ajouté à *BC*.

Données: *BC* - base de connaissances ONTOAST, *A*, *B* objets spatiaux définis *BC*
Sorties: $R_t \subseteq \mathcal{T}mp$

si $\exists! R \in \mathcal{T}mp : R(A,B) \in BC$ **alors**
  $R_t \leftarrow R$
  Return R_t
fin
si $\exists! R \in \mathcal{T}mp : R(B,A) \in BC$ **alors**
  $Rel \leftarrow R^{-1}$
  Return R_t
fin
si $\exists! R_1, R_2 \in \mathcal{T}mp : R_1(A,C) \in BC \wedge R_2(C,B) \in BC$ **alors**
  $R_t \leftarrow R_1 \circ R_2$
  $BC \leftarrow BC \cup R_t(A,B)$// `on ajoute à BC seulement les relations dont la déduction n'est pas immédiate`
  Return R_t
fin
si $knownTemp(A) \wedge knownTemp(B)$ **alors**
  $R_t \leftarrow computeTemporalTopology(Temp(A), Temp(B))$
  $BC \leftarrow BC \cup R_t(A,B)$
  Return R_t
fin
$CtNetwork \leftarrow buildCtNetwork(BC)$
$ArcConsistency(CtNetwork)$
$update(BC, CtNetwork)$
si $\exists R \in \mathcal{T}mp : R(A,B) \in BC$ **alors**
  $R_t \leftarrow \cup_{R \in \mathcal{T}mp : R(A,B) \in BC} R$ // $R_t \subseteq \mathcal{T}mp$
  Return R_t
fin
Return $\emptyset$ // `pas de solution`

Algorithme 3: TemporalRelation

6.5.3 Calcul de relations temporelles à partir des données quantitatives

Une relation temporelle peut être explicitement définie par l'utilisateur, mais également calculée. Pour calculer une relation temporelle à partir des données quantitatives, nous utilisons les opérateurs temporels définis par AROM-ST sur des données de type `interval`. Nous avons également ajouté au LMA de ONTOAST un nouvel opérateur, `temp` qui renvoie l'intervalle de temps attaché à un objet temporel donné comme paramètre (instance de `AROMTemporalConcept`). Ainsi, pour déterminer la relation temporelle qui est satisfaite par deux objets temporels A et B, nous utilisons l'arbre de décision présenté dans la figure 6.25.

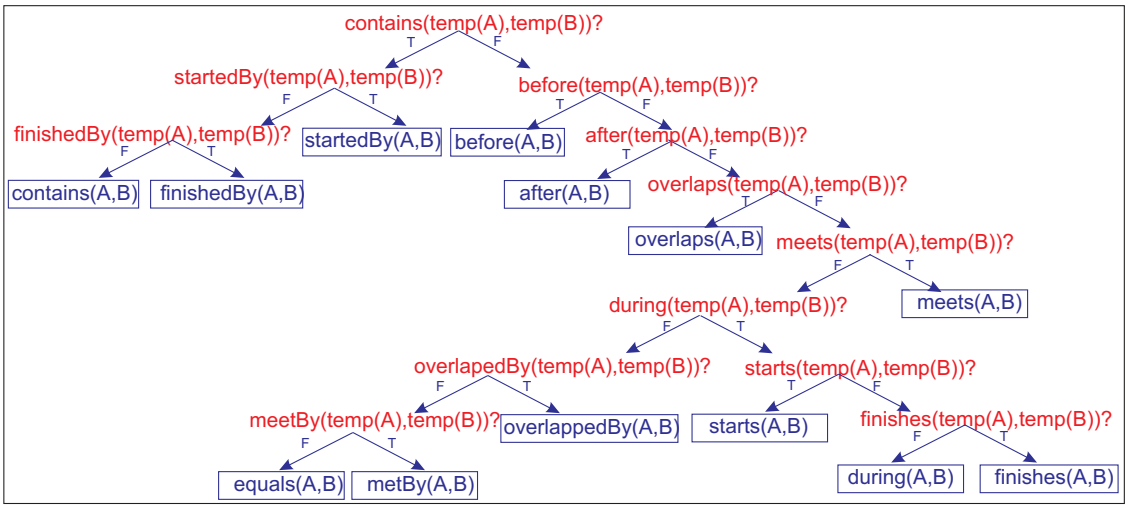


FIG. *6.25: Arbre de déduction de la relation temporelle satisfaite par deux objets temporels A et B. En non encadré les opérateurs temporels utilisés, et en encadré les relations temporelles qualitatives déduites.*

6.5.4 Déduction de relations temporelles à travers le raisonnement qualitatif

Les relations temporelles entre deux objets temporels A et B peuvent également être déduites à travers le raisonnement qualitatif. Pour cela, nous utilisons un réseau de contraintes qui modélise les relations temporelles comme des variables ayant comme domaine l'ensemble $\mathcal{T}mp$. Les objets temporels décrits dans la base de connaissances BC seront modélisés comme des constantes, regroupées dans l'ensemble $TemporalObjects$. Pour toute paire d'objets temporels (A,B), on définit une variable représentant la relation temporelle entre A et B comme suit :

$$\text{variable } Temp[A,B] \text{ où } Temp[A,B] \subseteq \mathcal{T}mp \text{ et } A,B \in TemporalObjects.$$

Le *modèle de contraintes temporelles* est constitué à partir des propriétés inter-relations définies par le tableau de composition des relations temporelles 2.10 (page 43), implémenté à l'aide de la fonction *compTemp*, et à partir du tableau de relations temporelles inverses 2.9 (page 43), implémenté par la fonction *convTemp* comme suit :

$$compTemp_{(A,B,C)} : Temp[A,B] \wedge Temp[B,C] \rightarrow Temp[A,C] \in compTemp(Temp[A,B], Temp[B,C]),$$
$$\forall A,B,C \in TemporalObjects \text{ et}$$

$$convTemp_{(A,B)} : Temp[A,B] \rightarrow Temp[B,A] \in convTemp(Temp[A,B]), \forall A,B \in TemporalObjects.$$

Pour les paires d'objets temporels (A,B) dont la relation temporelle $R_t \in 2^{\mathcal{T}mp}$ est explicitement contenue dans la base de connaissances, le domaine de la variable $Temp[A,B]$ est réduit à l'ensemble $\left\{R_{t_i} | R_t = \cup_{i=1}^{k} R_{t_i}\right\} \subseteq \mathcal{T}mp$. Pour les autres paires d'objets temporels, le domaine des variables *Temp* est fixé à $\mathcal{T}mp$. Une fois le réseau de contraintes construit, en utilisant un *solveur* qui impose l'arc consistance généralisée (GAC), ou qui cherche les solutions du *CSP*, on réduit autant que possible les domaines des variables *Temp*. Le réseau de contraintes ainsi filtré, sera utilisé pour mettre à jour la base de connaissances *BC*.

6.5.5 Ontologie de relations temporelles de référence pour ONTOAST

Pour faciliter la représentation des relations temporelles dans le cadre du Web Sémantique, l'ontologie `OWL-Time` a été proposée comme recommandation du W3C (voir section 3.4.3). Elle définit les 13 relations d'Allen que nous avons proposées d'introduire dans ONTAOST. Afin de garantir la cohérence de la traduction des descriptions temporelles de OWL vers ONTOAST, et de ONTOAST vers OWL, nous avons utilisé l'ontologie OWL-Time comme ontologie de référence. Ainsi, lors de l'importation d'une ontologie OWL, O_1, le parseur de ONTOAST peut construire des instances de relations temporelles seulement pour les descriptions réalisées par rapport à l'ontologie `OWL-Time`. Si l'ontologie O_1 contient des descriptions temporelles réalisées par rapport à une autre ontologie de relations temporelles, celles-ci ne sont pas traduites de façon adéquate, comme des tuples attachées aux relations temporelles prédéfinies de ONTOAST, mais comme des tuples de nouvelles associations. En conséquence, ces informations ne sont pas prises en compte pour l'inférence des nouvelles relations.

L'exportation des relations temporelles contenues par des bases de connaissances ONTOAST se fait par rapport à la même ontologie, `OWL-Time`.

Les déductions proposées par l'algorithme `TemporalRelation` (page 194) sont analogues à celles proposées par l'algorithme `SpatialRelations` (page 172).

6.6 Synthèse

Ce chapitre présente ONTOAST, une extension spatiale et temporelle de AROM-ONTO qui permet l'exploitation des descriptions spatiales et temporelles réalisées en OWL DL ou OWL 2 DL, à l'aide des types temporels proposés par les deux langages ainsi qu'à l'aide des concepts et des relations définies par les ontologies *GeoRSS-Simple* et *OWL-Time*. La présentation d'ONTOAST est organisée selon trois axes.

Dans un premier temps, nous avons présenté le modèle spatial et temporel quantitatif de ONTOAST, qui vient compléter le module AROM-ST proposé par Moisuc [174]. Nous avons ajouté de nouveaux types et opérateurs spatiaux et temporels à AROM-ST, et nous avons défini des règles de traduction des informations spatiales et temporelles quantitatives, décrites en utilisant les types proposés par AROM-ST, vers des descriptions OWL DL (ou OWL 2 DL). Ces traductions sont rapportées aux ontologies *GeoRSS-Simple* et *OWL-Time* que nous avons considérées comme ontologies de référence. Nous avons également défini des règles de traduction des descriptions ontologiques réalisées en OWL DL (ou OWL 2 DL) vers ONTOAST. Ces transformations sont nécessaires pour pouvoir utiliser les capacités de raisonnement de ONTOAST dans le cadre du Web Sémantique Géospatial.

Dans un deuxième temps, nous avons défini pour ONTOAST un modèle spatial et temporel qualitatif contenant un ensemble de *relations spatiales* : *topologiques*, de *direction* et de *distance* et un ensemble de *relations temporelles* pour lesquelles nous avons fixé la sémantique et le comportement. Ainsi, ces relations peuvent être instanciées sans redéfinition dans toute base de connaissances ONTOAST. Pour exploiter les informations spatiales/temporelles quantitatives ainsi que les relations spatiales/temporelles qualitatives décrites dans une base de connaissances ONTOAST, dans le but de trouver la relation spatiale/temporelle qui existe entre deux objets spatiaux/temporels *A* et *B*, nous avons défini un *algorithme de raisonnement spatial générique* et un *algorithme de raisonnement temporel*. Nous avons expliqué comment *l'algorithme spatial générique* peut être adapté pour prendre en compte les spécificités de chaque type de relation spatiale.

Le raisonnement quantitatif proposé par ONTOAST est basé sur l'exploitation d'un ensemble d'opérateurs spatiaux que nous ajoutons à AROM-ST.

Le raisonnement qualitatif est basé sur les tableaux des relations spatiales/temporelles inverses, et sur les tableaux de composition des relations spatiales/temporelles proposés par les approches de représentation qualitative de l'espace/du temps étudiées dans le chapitre 2. Ces descriptions axiomatiques des relations spatiales/temporelles sont utilisées pour la définition d'un réseau de contraintes (ou *CSP*). Le réseau est complété par l'ajout d'un ensemble de *variables topologiques*, de *direction* et de *distance* dont les domaines sont établis à partir des relations spatiales explicites qui existent dans la base de connaissances exploitée. Ce même type de réseau peut être construit pour les relations temporelles. En imposant l'arc consistance généralisé pour le réseau de contraintes et ensuite on cherchant ses solutions, les domaines des variables sont réduits autant que possible.

Suite à l'affinement de relations spatiales/temporelles à l'aide du *CSP*, la base de connaissances est mise à jour. La relation spatiale/temporelle qualitative, $rel, rel \in \{Topology, Direction, Distance, Temp\}$, cherchée pour la paire d'objets *A* et *B* est construite comme disjonction de l'ensemble d'éléments appartenant au domaine de la variable $rel[A,B]$ dans le *CSP*.

Dans un troisième temps, nous proposons une ontologie OWL DL pour la description des relations spatiales qualitatives, qui est compatible avec les relations gérées par ONTOAST. Cette ontologie peut être utilisée pour décrire des informations spatiales qui peuvent être exploitées par notre algorithme de raisonnement suite à leur traduction préalable en ONTOAST.

Dans ce sens, dans le chapitre suivant, nous présentons l'utilisation des capacités de raisonnement de ONTOAST pour la déduction des relations spatiales et temporelles implicites qui existent entre des individus décrits dans des ontologies OWL 2 DL.

Chapitre 7

Analyse Sémantique pour les ontologies OWL 2

... savoir analyser et reconstituer les cheminements de notre pensée, c'est nous mettre en mesure de les améliorer.

Seymour Papert

SOMMAIRE

Résumé

Dans ce chapitre nous proposons une adaptation de l'analyse sémantique pour les ontologies OWL DL ou OWL 2 DL. Notre approche entend mettre à profit la plus grande expressivité de ces deux langages par rapport au langage RDF(S) dans le processus de découverte d'associations sémantiques, qui explicite les relations implicites.

Nous proposons ici des algorithmes pour la découverte des associations sémantiques, qui prennent en compte le contexte thématique, spatial et temporel de la requête. Nous utilisons le raisonneur ONTOAST pour vérifier la cohérence entre les caractéristiques spatiales et temporelles des individus et des tuples étudiés, et les contextes spatial et temporel. Les capacités de raisonnement de ONTOAST sont également utilisées pour unifier dans un premier temps les relations spatiales et temporelles implicites qui relient les individus, qui peuvent être utilisées, dans un second temps, pour la découverte de nouvelles associations sémantiques.

7.1 Introduction

Dans la première partie de ce chapitre, nous présentons l'adaptation du concept d'*analyse sémantique*, initialement définie pour les graphes RDF(S) comme décrit dans la section 4.3.1, aux ontologies OWL 2 DL. Nous mettons en évidence l'intérêt de cette adaptation, ainsi que les difficultés que nous avons rencontrées lors de sa mise en œuvre.

Dans la deuxième partie du chapitre, nous proposons une technique de prise en compte de l'information spatiale et temporelle, utilisée pour décrire les ressources dans le cadre du Web Sémantique Géospatial. L'idée est de mettre en place un *framework* qui puisse gérer de façon native les informations spatiales et temporelles pour, d'une part, réduire le champ des recherches à une région de l'espace ou/et à un intervalle de temps, et, d'autre part, déduire davantage d'associations sémantiques, sur la base de proximités spatiale et/ou temporelle intéressantes entre individus.

Notre *framework* prend en compte les descriptions spatiales quantitatives conformes à l'ontologie *GeoRSS-Simple*, présentée dans la section 3.3.1, ainsi que les descriptions spatiales qualitatives conformément à l'ontologie *QualitativeSpatialRelations* proposée dans la section 3.3.2. Concernant l'information temporelle, le *framework* proposé dans ce chapitre (voir section 7.4) accepte les descriptions réalisées en utilisant les types temporels proposés par OWL 2 et celles conformes à l'ontologie OWL-Time.

Pour la gestion des informations spatiales et temporelles, nous utilisons le système ONTOAST (voir chapitre 6).

7.2 Associations sémantiques en OWL 2

Afin d'adapter la définition de l'*analyse sémantique* aux ontologies OWL 2 DL, nous rappelons la définition du vocabulaire et de l'interprétation d'une ontologie OWL 2.

7.2.1 Vocabulaire et interprétation des ontologies OWL 2

Le vocabulaire V utilisé par les ontologies OWL 2 est défini par rapport à un *Datatype Map* D (dont la spécification est donnée dans [180]) par le septuplet $V = (V_C, V_{OP}, V_{DP}, V_I, V_{DT}, V_{LT}, V_{FA})$, où :

- V_C est l'ensemble des classes contenues dans le vocabulaire V, qui inclut au moins les classes `owl:Thing` et `owl:Nothing` ;
- V_{OP} représente l'ensemble des propriétés-lien contenues par le vocabulaire V, qui inclut au moins les propriétés `owl:TopObjectProperty` et `owl:bottomObjectProperty` ;
- V_{DT} est l'ensemble d'attributs contenus par V, qui inclut au moins les propriétés `owl:topDataProperty` et `owl:bottomDataProperty` ;
- V_I est l'ensemble des individus, nommés ou anonymes, contenus par le vocabulaire V ;
- V_{DT} est un ensemble contenant tous les types de données de D, le type `rdfs:Literal` et éventuellement d'autres types de données : $N_{DT} \bigcup \{rdfs:Literal\} \subseteq V_{DT}$;
- la fonction V_{LT} associe un ensemble de littéraux, $LV\text{^^}DT$ à chaque type de données $DT \in N_{DT}$ et à chaque lexème $LV \in N_{LS}(DT)$;
- V_{FA} est une fonction qui associe un ensemble de paires (F, lt) pour chaque facette de restriction F, type de données $DT \in N_{DT}$ et littéral $lt \in V_{LT}$, tel que $\left(F, (LV, DT_1)^{LS}\right) \in N_{FS}(DT)$, où LV est la forme lexicale de lt et DT_1 est le type de lt.

La même source [180] définit une interprétation OWL 2 par rapport à un *Datatype Map D* et un vocabulaire V comme étant le 9-uplet $I = \left(\Delta_I, \Delta_D, \bullet^C, \bullet^{OP}, \bullet^{DP}, \bullet^I, \bullet^{DT}, \bullet^{LT}, \bullet^{FA}\right)$, où :

- Δ_I est un ensemble non vide appelé domaine abstrait (ou domaine d'individus) ;
- Δ_D est un ensemble non vide appelé domaine concret, tel que $(DT)^{DT} \subseteq \Delta_D, \forall DT \in V_{DT}$. Les ensembles Δ_D et Δ_I sont disjoints.
- $\bullet^C$ est la fonction d'interprétation des classes, qui associe à chaque classe $C \in V_C$ un sous-ensemble $(C)^C \subseteq \Delta_I$, tel que $\left(\texttt{owl:Thing}^C\right) = \Delta_I$ et $(\texttt{owl:Nothing})^C = \emptyset$;
- $\bullet^{OP}$ est la fonction d'interprétation des propriétés-lien qui associe à chaque propriété-lien $OP \in V_{OP}$ un sous-ensemble $(OP)^{OP} \subseteq \Delta_I \times \Delta_I$ tel que $(\texttt{owl:topObjectProperty})^{OP} = \Delta_I \times \Delta_I$ et $(\texttt{owl:bottomObjectProperty})^{OP} = \emptyset$;
- $\bullet^{DP}$ est la fonction d'interprétation des attributs qui associe à chaque propriété concrète $DP \in V_{DP}$ un sous-ensemble $(DP)^{DP} \subseteq \Delta_I \times \Delta_D$ tel que $(\texttt{owl:topDataProperty})^{DP} = \Delta_I \times \Delta_D$ et $(\texttt{owl:bottomDataProperty})^{DP} = \emptyset$;
- $\bullet^I$ est la fonction d'interprétation des individus, qui associe à chaque individu $a \in V_I$ un élément $(a)^I \in \Delta_I$;
- $\bullet^{DT}$ est la fonction d'interprétation des types de données, qui associe à chaque type de données $DT \in V_{DT}$ un sous-ensemble $(DT)^{DT} \subseteq \Delta_D$, tel que :
  - $\bullet^{DT}$ est équivalente à la fonction homonyme définie par D pour chaque type de données $DT \in V_{DT}$ et
  - $(rdfs:Literal)^{DT} = \Delta_D$.
- $\bullet^{LT}$ est la fonction d'interprétation des littéraux définie par $(lt)^{LT} = (LV, DT)^{LS}$ pour chaque $lt \in V_{LT}$, où LV est la forme lexicale de lt, et DT est le type de lt ;
- $\bullet^{FA}$ est la fonction d'interprétation des facettes de restriction définie comme $(F, lt)^{FA} = \left(F, (lt)^{LT}\right)^{FS}$ pour chaque $(F, lt) \in V_{FA}$.

Notation 7.1. *Par $Axiom(O)$, on dénote l'ensemble d'axiomes définis dans la fermeture transitive de l'ontologie O.*

Notation 7.2. *Par $inverseOf(P)$ on dénote l'expression à base de propriétés $\texttt{InverseObjectProperty}(P)$, où $P \in V_{OP}$ est une propriété-lien définie dans la fermeture transitive de l'ontologie O.*

7.2.2 Séquences de propriétés et $onto - paths$ en OWL 2

Dans cette section, nous proposons d'étendre la notion d'*association sémantique* définie par [18, 219, 19, 124, 218, 199] pour les graphes RDF(S) afin qu'elle puisse prendre en compte et exploiter des descriptions réalisées par rapport aux ontologies OWL 2. Notre travail est motivé par les limitations de l'analyse sémantique présentées dans la section 4.3.6. Nous précisons qu'étant donnée la compatibilité descendante[1] du langage OWL 2 avec les langages OWL et RDF(S), notre approche peut être utilisée pour chacun des trois types de d'ontologies : OWL 2, OWL et RDF(S).

Notation 7.3. *Dans ce chapitre, nous utilisons les fonctions **domain** et **range** pour dénoter l'ensemble d'expressions à base de classes qui forment respectivement le domaine et l'image de l'expression à base de propriétés-lien, OPE, passée en paramètre :*

$\textbf{\textit{domain}}(OPE) =_{def} \{CE_i | ObjectPropertyDomain(OPE\ CE_i) \in Axiom(O)\}$,

[1] En anglais *backwards compatibility*.

$\forall a,b \in \Delta_I : (a,b) \in (OPE)^{OP} \rightarrow \exists CE_a \in domain(OPE) : a \in (CE_a)^C$
$\boldsymbol{range}(OPE) =_{def} \{CE_i | ObjectPropertyRange(OPE\ CE_i) \in Axiom(O)\}$,
$\forall a,b \in \Delta_I : (a,b) \in (OPE)^{OP} \rightarrow \exists CE_b \in range(OPE) : b \in (CE_b)^C$.

Notation 7.4. *Par $\overline{domain}(OPE)$ et $\overline{range}(OPE)$ on dénote l'expression à base de classes définie comme étant la disjonction de toutes les expressions à base de classes incluses par respectivement le domaine et l'image de l'expression à base de propriétés-lien, OPE, passée en paramètre :*
$\overline{domain}(OPE) =_{def} \sqcup_{CE_i \in domain(OPE)} CE_i$ *et*
$\overline{range}(OPE) =_{def} \sqcup_{CE_i \in range(OPE)} CE_i$ *et*

À titre d'exemple, considérons les deux extraits d'ontologie présentées dans la figure 7.1. Pour l'expression à base de propriétés `paints` (voir figure 7.1a), les fonctions ***domain*** et ***range*** renvoient les ensembles de classes {`Painter`} respectivement {`Painting`}, et les fonctions $\overline{domain}$ et $\overline{range}$ renvoient les expressions à base de classes `Painter` et respectivement `Painting`. Dans ce cas, $\overline{domain}$ et $\overline{range}$ représentent des classes simples car le domaine et l'image de la propriété visée sont simples. En revanche, pour l'expression à base de classes {`sculpts`} (voir figure 7.1b) la fonction ***domain*** renvoie l'ensemble {`Artist`, ¬`Painter`}, alors que la fonction $\overline{domain}$ a comme résultat l'expression à base de classes `Artist` ⊔ ¬`Painter` (la disjonction des expressions à base de classes appartenant à l'ensemble ***domain***).

```
<ObjectPropertyDomain>
    <ObjectProperty URI="art#paints"/>
    <Class URI="art#Painter"/>
</ObjectPropertyDomain>
<ObjectPropertyRange>
    <ObjectProperty URI="art#paints"/>
    <Class URI="art#Painting"/>
</ObjectPropertyRange>
```

(a) Exemple de domaine simple.

```
ObjectPropertyDomain>
    <ObjectProperty URI="art#sculpts"/>
    <ObjectIntersectionOf>
        <Class URI="art#Artist"/>
        <ObjectComplementOf>
            <Class URI="art#Painter"/>
        </ObjectComplementOf>
    </ObjectIntersectionOf>
</ObjectPropertyDomain>
<ObjectPropertyRange>
    <ObjectProperty URI="art#sculpts"/>
    <Class URI="art#Sculpture"/>
</ObjectPropertyRange>
```

(b) Exemple de domaine défini à l'aide d'une expression à base de classes.

FIG. *7.1: Extrait d'une ontologie qui définit deux propriétés en OWL 2.*

Par la suite, nous présentons l'évolution que nous proposons pour chacune des notions utilisées par l'analyse sémantique. Nous donnons des exemples qui montrent l'utilité de ces adaptations.

Définition 7.1. *Une séquence de propriétés OWL 2 est un ensemble fini et ordonné d'expressions à base de propriétés-lien, définies dans la fermeture transitive d'une ontologie O, appelée ontologie de référence. Une séquence de propriété OWL 2 est définie formellement par :*

$$PS =_{def} \lfloor OPE_1 \bullet OPE_2 \bullet \ldots \bullet OPE_n \rfloor,$$

tel que les deux conditions suivantes soient respectées :

a) $\forall i \in 1..n, (OPE_i = P_i) \vee (OPE_i = \texttt{InverseObjectProperty}(P_i)), P_i \in V_{OP}$;

b) $\forall i \in 1..n : (\overline{range}(OPE_{i-1}))^C \cap (\overline{domain}(OPE_i))^C \neq \emptyset$.

Définition 7.2. *Une séquence de propriétés est réalisable par rapport à une ontologie O et un Datatype Map D, si et seulement si il existe un modèle I de O par rapport à D et V pour lequel le*

domaine abstrait Δ_I contienne au moins une séquence d'individus distincts $o_i \in \Delta_I, i \in 1..n$, tels que $\langle o_{i-1}, o_i \rangle \in (OPE_i)^{OP}, \forall i \in 1..n$. Dans ce cas, on dit que la séquence de propriétés PS est satisfaite par l'interprétation I, et qu'elle a au moins une instance ps, $ps \in (PS)^I \subseteq \times_{i=1}^{n} (OPE_i)^{OP}$.

Une instance d'une séquence de propriétés *PS* est appelée $onto-path$. Pour une description non ambiguë des $onto-paths$, nous utilisons la notation suivante :

$$ps = x \overset{OPE_1}{\rightarrow} o_1 \overset{OPE_2}{\rightarrow} \ldots \overset{OPE_{n-1}}{\rightarrow} o_{n-1} \overset{OPE_n}{\rightarrow} y,$$
$$o_i \in \Delta_I, \forall i \in 0..n \wedge x = o_0, y = o_n \wedge \langle o_{j-1}, o_j \rangle \in (OPE_j)^{OP}, \forall j \in 1..n. \quad (7.1)$$

Nous rappelons que l'objet *x* est appelé *origine* de l'$onto-path$, tandis que l'objet *y* est appelé *terminus*. Ils sont désignés à l'aide des fonctions $origine(ps)$ et $terminus(ps)$ respectivement.

L'évolution principale proposée par la définition 7.1, par rapport à la définition 4.1, vient de l'utilisation des *expressions à base de propriétés-lien* à la place des propriétés simples de RDF(S), pour la construction des séquences de propriétés. Une expression à base de propriétés-lien est définie en OWL 2 comme étant, soit une *propriété-lien*, soit *l'inverse d'une propriété-lien*. En conséquence, en OWL 2, chaque propriété a une inverse, même si celle-ci n'est pas explicitement définie.

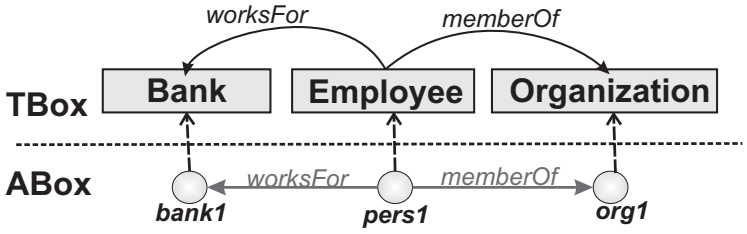


FIG. *7.2: Ontologie simple modélisant un domaine lié au terrorisme.*

L'utilisation des *expressions à base de propriétés* dans la construction des séquences de propriétés, contribue à la découverte de nouveaux *onto-paths*. Considérons, par exemple, l'ontologie O_1 présentée dans la figure 7.2, dans laquelle les classes `Bank` et `Employee` sont définies comme étant disjointes. En utilisant la définition 4.1, on peut exclusivement construire les séquences de propriétés PS_1 et PS_2, où

$PS_1 = \lfloor \texttt{worksFor} \bullet \texttt{memberOf} \rfloor$ et
$PS_2 = \lfloor \texttt{memberOf} \bullet \texttt{worksFor} \rfloor$.

Celles-ci sont toutes les deux irréalisables par rapport à l'ontologie O_1. En revanche, la définition 7.1, que nous proposons interdit la construction des séquences PS_1 et PS_2, mais facilite la construction de deux autres séquences de propriétés, PS_3 et PS_4, où :

$PS_3 = \lfloor inverseOf(\texttt{worksFor}) \bullet \texttt{memberOf} \rfloor$ et
$PS_4 = \lfloor inverseOf(\texttt{memberOf}) \bullet \texttt{worksFor} \rfloor$.

Les nouvelles séquences de propriétés, PS_3 et PS_4 sont réalisables dans l'ontologie O_1 et elles possèdent les instances ps_3 et respectivement ps_4, où :

$ps_3 = bank_1 \overset{inverseOf(worksFor)}{\rightarrow} pers_1 \overset{memberOf}{\rightarrow} org_1, ps_3 \in (PS_3)^I$ et
$ps_4 = org_1 \overset{inverseOf(memberOf)}{\rightarrow} pers_1 \overset{worksFor}{\rightarrow} bank_1, ps_4 \in (PS_4)^I$.

Par rapport à la définition 4.1, la définition que nous proposons ajoute une nouvelle restriction qui doit être satisfaite lors de la construction des séquences de propriétés (voir condition *b)* dans la définition 7.1). Celle-ci exige que l'image de chaque expression à base de propriétés, *OPE*, incluse dans une séquence de propriétés, *PS*, partage au moins une instance avec le domaine de l'expression à base de propriétés qui la suit dans *PS*. Cette condition garantit la construction exclusive de séquences de propriétés correctes, ou, en d'autres termes, la construction de séquences de propriétés qui peuvent posséder des instances. Cette condition considère donc dans l'exemple précedent, les séquences de propriétés PS_1 et PS_2 comme étant invalides. L'invalidité de PS_1 vient du fait que l'image de la propriété-lien `worksFor` ne partage pas d'instance avec le domaine de la propriété `memberOf`, il est donc impossible de trouver un objet intermédiaire $a \in \Delta_I$ tel que $\langle x,a\rangle \in (\texttt{worksFor})^{OP} \wedge \langle a,y\rangle \in (\texttt{memberOf})^{OP}, \forall x \in \left(\overline{domain}\,(\texttt{worksFor})\right)^{C}$, $\forall y \in (\overline{range}\,(\texttt{memberOf}))^{C}$. L'invalidité de PS_2 est déduite d'un raisonnement analogue.

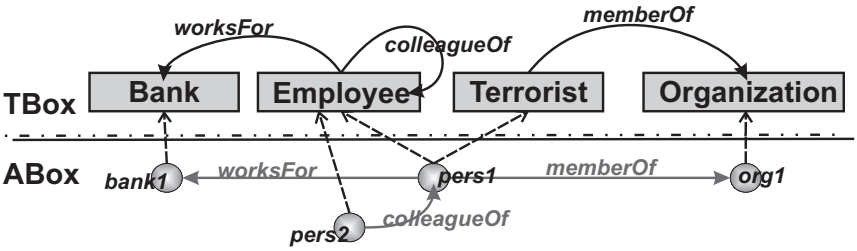


FIG. *7.3: Ontologie simple modélisant un domaine lié au terrorisme.*

Il est également important de souligner que la définition que nous proposons pour les séquences de propriétés couvre les situations inhabituelles, pour lesquelles, à première vue, la construction d'une séquence de propriétés n'est pas possible. Ce genre de situation est due à la classification multiple des ressources autorisée par le modèle de données OWL. Par exemple, dans l'ontologie illustrée dans la figure 7.3, en examinant exclusivement la TBox, on a tendance à conclure que les seules séquences de propriétés qui existent sont :

$S_1 = \lfloor \texttt{memberOf} \rfloor$,
$S_2 = \lfloor inverseOf(\texttt{memberOf}) \rfloor$,
$S_3 = \lfloor \texttt{colleagueOf} \rfloor$,
$S_4 = \lfloor inverseOf(\texttt{colleagueOf}) \rfloor$,
$S_5 = \lfloor \texttt{worksFor} \rfloor$,
$S_6 = \lfloor inverseOf(\texttt{worksFor}) \rfloor$,
$S_7 = \lfloor inverseOf(\texttt{worksFor}) \bullet \texttt{colleagueOf} \rfloor$ et
$S_8 = \lfloor \texttt{colleagueOf} \bullet \texttt{worksFor} \rfloor$.

Or, en examinant la ABox, on se rend compte de l'existence d'un $onto-path$ qui ne correspond à aucune des séquences de propriétés décrites auparavant. Il s'agit de l'$onto-path$

$p_1 = bank_1 \overset{inverseOf(worksFor)}{\longrightarrow} pers_1 \overset{memberOf}{\longrightarrow} org_1$.

Celui-ci est attaché à la séquence de propriétés

$S_9 = \lfloor inverseOf(\texttt{worksFor}) \bullet \texttt{memberOf} \rfloor$.

Cette séquence de propriétés est valide, car l'image de l'expression à base de propriétés $inverseOf(\texttt{worksFor})$ partage une instance avec le domaine de la propriété `memberOf`. Ceci explique notre choix de définir la validité des séquences de propriétés, non pas en fonction des

expressions à base de classes définies comme *domaine* et *image* des propriétés incluses par les séquences respectives, mais en fonction de l'interprétation de celles-ci.

Proposition 7.1. *Pour chaque séquence de propriétés* $PS =_{def} \lfloor OPE_1 \bullet OPE_2 \bullet \ldots \bullet OPE_n \rfloor$*, définie par rapport à une ontologie OWL 2 de référence O, on peut construire une unique séquence de propriétés inverses comme suit :*

$PS^{-1} = inverseProp(OPE_n) \bullet inverseProp(OPE_{n-1}) \bullet \ldots \bullet inverseProp(OPE_1)$, *où :* $\forall OPE$

$$inverseProp(OPE) = \begin{cases} P_i, & OPE = InverseObjectProperty(P_i) \\ inverseOf(P_i), & OPE = P_i \end{cases}, P_i \in V_{OP}.$$

La démonstration de la proposition 7.1 est immédiate. À partir de la définition de la fonction *inverseProp*, on déduit que la séquence de propriétés PS^{-1} est construite en remplaçant chaque expression à base de propriétés-lien OPE_i incluse dans PS par son inverse. Comme l'ontologie de référence est une ontologie OWL 2, toute expression à base de propriétés définie dans O possède une inverse. L'unicité de la séquence de propriétés PS^{-1} ainsi construite est garantie par l'ordre respecté par les expressions à base de propriétés qu'elle inclut. Concrètement, PS^{-1} contient les inverses des expressions à base de propriétés incluses dans PS, qui sont prises dans l'ordre inverse de leur positions dans PS. De plus, la séquence de propriétés PS^{-1} est correcte car, $\forall i \in n..2$, on a :

$range(inverseProp(OPE_i)) = domain(OPE_i)$ (définition de la fonction *inverseProp*)
$\wedge\ domain(inverseProp(OPE_i)) = range(OPE_i)$ (définition de la fonction *inverseProp*)
$\wedge$ PS est correcte $\Rightarrow \overline{domain}(OPE_i) \cap \overline{range}(OPE_{i-1}) \neq \emptyset$ (définition 7.1)
$\Rightarrow \overline{range}(inverseProp(OPE_i)) \cap \overline{domain}(inverseProp(OPE_{i-1})) \neq \emptyset$.

Cette propriété sera utilisée pour la déduction des *associations sémantiques* reliant deux individus x et y. Concrètement, en utilisant l'analyse sémantique des graphes RDF(S), pour trouver toutes les associations sémantiques entre x et y, on doit construire deux types de séquences de propriétés :

- l'un qui relie chacune des classes auxquelles l'individu x est attaché à chacune des classes dont l'individu y est une instance, et
- le deuxième qui relie chaque classe à laquelle y est attaché à chaque classe dont x est une instance.

Cela est dû au fait que les propriétés RDF(S) ne permettent pas d'être parcourues dans les deux sens. En revanche, pour la déduction des associations sémantiques entre x et y par rapport à une ontologie OWL 2, il suffit de construire les séquences de propriétés dans un seul sens (*i.e.* de la classe de x vers la classe de y ou l'inverse). Les séquences de propriétés inverses peuvent être déduites (si besoin) en utilisant la proposition 7.1.

Afin d'illustrer nos idées, par la suite, nous considérons l'ontologie simple présentée dans la figure 7.3. Celle-ci décrit un domaine lié au terrorisme, qui prend en compte quatre concepts : `Terrorist`, `Organization`, `Employee` et `Bank` auxquels s'ajoutent les relations `worksFor`, `colleagueOf` et `memberOf`. Dans cette ontologie, on peut identifier au moins deux séquences de propriétés : $PS_1 = \lfloor$`memberOf`$\rfloor$ et $PS_2 = \lfloor$`colleagueOf`$\bullet$`worksFor`$\rfloor$, décrivant respectivement les liens possibles entre les terroristes (`Terrorist`) et les différents organisations (`Organization`), et entre les employés (`Employee`) et les banques où ils travaillent (`Bank`). Les séquences PS_1 et PS_2 sont toutes les deux satisfaites, car elles ont au moins une instance chacune, représentant des *associations sémantiques* entre $pers_1$ et org_1, respectivement $pers_2$ et $bank_1$:

$ps_1 = pers_1 \overset{memberOf}{\rightarrow} org_1,\ ps_1 \in (PS_1)^I$

$ps_2 = pers_2 \overset{colleagueOf}{\rightarrow} pers_1 \overset{worksFor}{\rightarrow} bank_1,\ ps_2 \in (PS_2)^I$.

La fonction $NodesOfPS$ adaptée pour les ontologies OWL 2 renvoie $NodesOfPS(PS) = \{CE_1, CE_2, \ldots, CE_k\}$ tel que $\forall i \in 1..k, \exists j \in 1..n : CE_i \in domaine(PE_j) \vee CE_i \in image(PE_j), PE_j \in PS$.

Pour la séquence de propriétés S_9, la fonction $NodesOfPS$ renvoie l'ensemble d'expressions à base de classes

$NodesOfPS(S_9) = \{\texttt{Bank}, \texttt{Employee}, \texttt{Employee} \sqcap \texttt{Terrorist}, \texttt{Organization}\}$. Pour faciliter la manipulation des objets inclus dans les $onto-paths$, nous avons adapté la fonction $PathNodes$ pour les ontologies OWL 2, comme illustré par l'équation 7.2. Pour les $onto-paths$ ps_1 et ps_2 cette fonction renvoie les ensembles $PathNodes(ps_1) = \{pers_1, org_1\}$ et $PathNodes(ps_2) = \{pers_2, pers_1, bank_1\}$.

$$PathNodes(ps) = \left\{ o_0, o_1, \ldots, o_n | o_i \in \Delta_I, i \in 0..n \wedge \langle o_{j-1}, o_j \rangle \in (OPE_j)^{OP}, \forall j \in 1..n \right\} \quad (7.2)$$

.

Proposition 7.2. *Une séquence de propriétés PS, est impliquée par une ontologie O si et seulement si toute interprétation I de O par rapport à D et V satisfait PS. Dans ce cas, on dit que PS est une conséquence logique de l'ontologie O : $O \models PS$.*

7.2.3 Associations sémantiques

Dans le chapitre 4, nous avons présenté trois types d'associations sémantiques : $\rho - pathAssociated$, $\rho - isoAssociated$ et $\rho - jointAssociated$. Néanmoins, dans le cadre de cette thèse, nous nous concentrons exclusivement sur l'adaptation pour OWL 2, des associations de type $\rho - pathAssociated$. Les associations de type $\rho - isoAssociated$ et $\rho - jointAssociated$, bien que très intéressantes, en particulier dans des domaines où les dimensions spatiale et temporelle sont importantes (tels que le tourisme ou le transport), sont également bien plus complexes que les associations de type $\rho - pathAssociated$. Le cas de ces assocaitions sera toutefois exploré dans un travail futur (voir chapitre 8).

La notion d'*association sémantique* est construite à partir du concept de *séquence de propriétés* défini dans la section précédente.

Proposition 7.3. *Deux individus x et y, décrits dans la fermeture transitive d'une ontologie O, ont une association sémantique de type ontoPathAssociated, si et seulement s'il existe au moins une séquence de propriétés PS qui est satisfaite dans un modèle I de O :* $\exists ps \in (PS)^I$ *tel que* $origine(ps) = x$ *et* $terminus(ps) = y$ *ou l'inverse. Dans ce cas, on dit que ps satisfait la relation* $ontoPathAssociated(x,y)$ *:* $ps \models$ ***ontoPathAssociated***(x,y).

Si, entre deux individus x et y, il existe une relation de type $ontoPathAssociated$, alors dans la partie assertionnelle de la fermeture transitive de l'ontologie de référence O, il existe un chemin qui relie directement x à y, ou en passant par des individus intermédiaires.

Proposition 7.4. *Deux entités x et y, décrites dans une ontologie O, ont une association sémantique dans une interprétation I de O, si et seulement si la relation* $ontoPathAssociated(x,y)$ *est satisfaite dans I.*

7.2.4 Requêtes sémantiques

Une requête sémantique est définie en utilisant l'opérateur binaire *ontoPathAssociated*, et a comme paramètres les individus x et y pour lesquels on désire trouver les associations sémantiques. Les résultats d'une requête sémantique forment un ensemble de $onto-paths$ p_i tels que : $p_i \models ontoPathAssociated(x,y)$.

7.2.5 Contexte thématique d'une requête sémantique

Le contexte thématique d'une requête sémantique capture le domaine d'intérêt de l'utilisateur et sert à focaliser la recherche des associations sémantiques sur des concepts et des relations qui sont considérés comme discriminants pour la requête courante. Autrement dit, le contexte thématique sert à choisir, parmi l'ensemble de réponses possibles, les associations sémantiques qui sont considérées comme prioritaire par l'utilisateur. Concrètement, lors de la définition du contexte thématique, Ct_{Them}, une ou plusieurs régions d'intérêt $Ct_1, Ct_2, \ldots, Ct_l$ sont décrites sous forme d'ensembles, non nécessairement disjoints, contenant à la fois des expressions à base de classes et des expressions à base de propriétés. Si une expression à base de classes, ou une expression à base de propriétés, est incluse dans une région d'intérêt Ct_i, toutes ses sous-classes, et toutes ses sous-propriétés, sont considérées comme faisant partie de la région d'intérêt Ct_i.

Comme nous l'avons décrit dans la section 4.3.4, à chaque région d'intérêt Ct_i peut être attaché un poids $w_i, w_i \in [0,1]$, qui quantifie son importance. Les poids sont utilisés dans le processus de classification des résultats afin de calculer le rang des associations sémantiques dans la liste des résultats.

L'intérêt de l'adaptation que nous proposons pour la notion de contexte thématique, par rapport au contexte utilisé par [219, 19, 124], vient de la flexibilité offerte par OWL 2 en ce qui concerne la définition des expressions à base de classes. Concrètement, alors que le contexte thématique proposé par [219, 19, 124] peut contenir exclusivement des classes et des propriétés RDF(S) simples, la notion de contexte que nous proposons est beaucoup plus souple. Celui-ci peut inclure toute expression logique construite à partir des expressions à base de classes décrites dans l'ontologie de référence. Par exemple, on peut définir une région contextuelle $Ct_1 = \{A \sqcap B \sqcup \{o_1, o_2\}, \neg C, \ldots\}$. Dans ce cas, chaque élément Exp de Ct_1 (expression à base de classes ou expression à base de propriétés) est classé par rapport à la hiérarchie de classes et de propriétés, définie par l'ontologie de référence. L'objectif est de déduire les expressions à base de classes/propriétés subsumées par Exp, et ainsi de construire les régions d'intérêt.

Nous proposons l'introduction d'un paramètre de contexte : la *déviation*, *CtDev*. Celle-ci est une valeur entière qui précise le nombre maximum d'éléments successifs (individus ou instances d'expressions à base de propriétés-lien) qui peuvent être en dehors (donc dans le voisinage) de toute région d'intérêt. Par exemple, une *déviation* contextuelle égale à 3 impose qu'un $onto-path$, valide par rapport à un contexte Ct_{Them}, ne peut pas contenir plus de 3 éléments successifs qui ne sont pas inclus dans une région d'intérêt de Ct_{Them}. L'intérêt de définir une déviation contextuelle par rapport à la possibilité de simplement définir un contexte plus large, vient du fait qu'il est très difficile de savoir *a priori* quels types d'instances et de tuples seront incluses par les $onto-paths$ cherchés.

Ainsi, le contexte thématique peut être considéré comme un filtre qui limite la découverte des associations sémantiques aux régions d'intérêt, ainsi qu'à leur voisinage immédiat.

Pour illustrer notre démarche, considérons l'extrait d'ontologie présenté dans la figure 7.4. Pour plus de lisibilité, nous avons omis les liens d'instanciation entre les éléments de la ABox et ceux de la TBox. Néanmoins, ces relations peuvent être ici déduites à partir des noms choisis pour les individus. Dans cette ontologie de référence, il s'agit de trouver les associations sémantiques qui existent entre les individus $pers_1$ et $pers_3$. La requête $ontoPathAssociated(pers_1, pers_3)$ est donc formulée. Sachant que l'on s'intéresse exclusivement à des personnes qui sont membres d'au moins une organisation terroriste et aux organisations impliquées dans l'attaque $attack_1$, nous définissons le contexte $Ct_{TerrAtt} = \{\exists$`memberOf TerroristOrg`$, \exists$`involvedIn` $\{$`attack`$_1\}$, `memberOf`$\}$ et la déviation $CtDev = 2$. En d'autres termes, à travers le contexte $Ct_{TerrAtt}$, la recherche est concentrée sur :

- les personnes qui sont membres d'une organisation terroriste,
- les organisations qui sont impliquées dans l'attaque terroriste $attack_1$
- toute relation `memberOf` ainsi que son inverse.

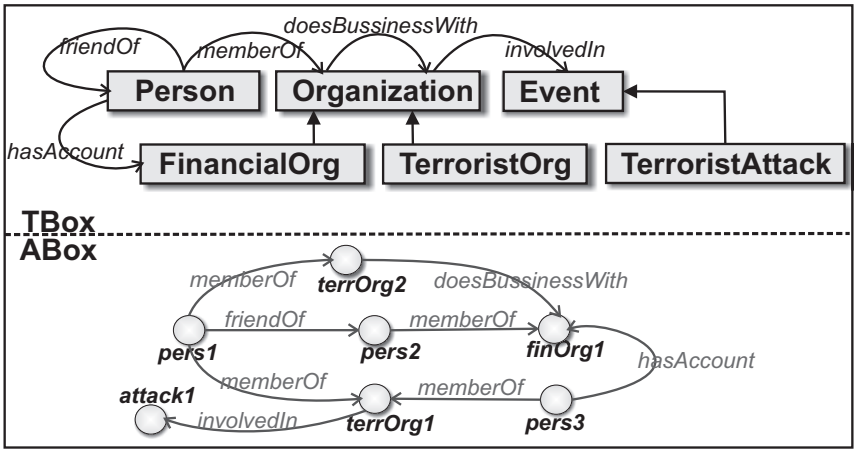


FIG. *7.4: Illustration d'un extrait de l'ontologie OWL 2 DL qui traduit le graphe RDF(S) présenté dans la figure 4.4. Celui-ci propose une modélisation du domaine de la sécurité d'un territoire.*

TAB. *7.1:* Ontology-paths *qui connectent les individus $pers_1$ et $pers_3$ de l'ontologie de la figure 7.4.*

$p_1 = pers_1 \overset{memberOf}{\rightarrow} terrOrg_2 \overset{doesBussinessWith}{\rightarrow} finOrg_1 \overset{inverseOf(hasAccount)}{\rightarrow} pers_3$
$p_2 = pers_1 \overset{friendOf}{\rightarrow} pers_2 \overset{memberOf}{\rightarrow} finOrg_1 \overset{inverseOf(hasAccount)}{\rightarrow} pers_3$
$p_3 = pers_1 \overset{memberOf}{\rightarrow} terrOrg1 \overset{inverseOf(memberOf)}{\rightarrow} pers_3$

Si on ignore le contexte, pour la requête $ontoAssociated(pers_1, pers_3)$, on trouve les solutions illustrées dans le tableau 7.1. Ces résultats peuvent être affinés en utilisant les informations contextuelles. Concrètement, l'$onto-path$ p_1 est éliminé de l'ensemble des résultats, car il contient trois éléments successifs qui ne font pas partie de $Ct_{TerrAtt}$: `doesBussinessWith`, `finOrg`$_1$ et `hasAccount`, alors que la déviation maximale admise est 2. L'$onto-path$ p_2 satisfait le contexte $Ct_{TerrAtt}$ car $pers_1 \in (\exists$`memberOf TerroristOrg`$)^C$, le tuple `friendOf`$(pers_1, pers_2)$ et l'instance $pers_2$ ne font pas partie du contexte mais sont suivis par le tuple `memberOf`$(pers_2, finOrg_1)$, qui est considéré intéressant pour la requête courante. De même, l'instance $finOrg_1$, et le tuple $inverseOf($`hasAccount`$(finOrg_1, pers_3))$ ne sont pas inclus dans $Ct_{TerrAtt}$, mais sont suivis par

$pers_3$ qui l'est (car $pers_3 \in (\exists \texttt{memberOfTerroristOrg})$). La longueur de la séquence de déviation est encore une fois égale à *CtDev* et le résultat p_2 est valide. De façon analogue, on valide l'$onto - path$ p_3.

7.3 Associations sémantiques spatio-temporelles

Dans cette section, nous présentons une technique de gestion des informations spatiales et temporelles attachées aux ressources décrites dans des ontologies OWL 2, lors de l'analyse sémantique.

7.3.1 Descriptions spatiales prises en charge

Notre système prend en charge les descriptions spatiales conformes à l'ontologie *GeoRSS Simple*, présentée dans la section 3.3.1, et à l'ontologie *QualitativeSpatialRelation*, proposée dans la section 6.18. Ce choix est motivé par le fait que ces deux ontologies sont compatibles avec le raisonneur ONTOAST. Notre idée est de traduire dans ONTOAST les descriptions spatiales contenues dans la fermeture transitive de l'ontologie de référence, afin qu'elles puissent être exploitées en utilisant notre raisonneur. Les informations spatiales sont utiles, d'une part, dans le processus de filtrage des connaissances par rapport à un contexte spatial donné, et, d'autre part, pour déduire de nouvelles relations spatiales qui peuvent nous amener à découvrir de nouvelles connexions entre individus.

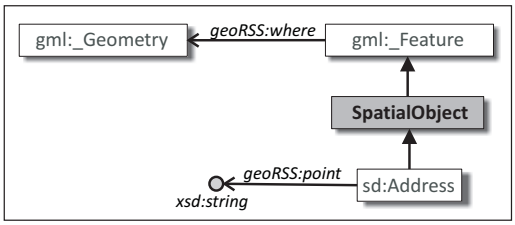


FIG. *7.5: L'ontologie sd (acronyme de spatial description) qui décrit les descriptions spatiales prises en compte par l'analyse sémantique spatiale.*

Comme illustré par la figure 7.5, les objets disposant d'une description spatiale sont des instances de la classe `SpatialObject`. Celle-ci est une spécialisation de la classe `gml:_Feature` utilisée dans l'ontologie *GeoRSS-Simple* comme domaine d'un ensemble d'attributs, utiles pour décrire les caractéristiques spatiales des éléments (voir section 3.3.1). L'emprise spatiale d'un objet peut également être définie à travers la relation `qsr:where` qui désigne une description géométrique, instance de la classe `gml:_Geometry`. Une adresse est un objet spatial particulier, car elle peut être facilement transformée en une paire de coordonnées en utilisant un service Web dédié, tel que *Google Maps*. Le résultat d'une telle transformation est représenté comme valeur de type `xsd:string`, désignée par l'attribut `geoRSS:point` (voir figure 7.5).

Un objet spatial peut également être décrit en utilisant les relations spatiales définies dans l'ontologie `QualitativeSpatialRelations` (voir section 6.18).

7.3.2 Contexte spatial d'une requête sémantique

Les capacités d'inférence spatiale offertes par ONTOAST peuvent être utilisées pour filtrer les connaissances prises en compte lors de la recherche des $onto-paths$ entre deux objets x et y. Pour cela, on définit $C_{Spatial}$, le contexte spatial d'une requête, comme une ou plusieurs paires $(so, relSp)$, où $so \in (SpatialObject)^C$, et $relSp$ est l'une des relations spatiales qualitatives définies par l'ontologie `QualitativeSpatialRelations`. Comme exposé dans la section 6.18, les relations spatiales que nous avons définies ne sont pas modélisées de façon uniforme : les relations topologiques et les relations de direction de type carreau unique sont représentées comme des propriétés-lien, tandis que les relations de distance et les relations de direction de type carreaux multiples, sont réifiées par des instances de classes. Pour cette raison, une séparation du contexte spatial s'impose, telle que $C_{Spatial} = C_{TopDir} \cup C_{DirDist}$. Le premier sous-contexte, C_{TopDir}, est spécifié en utilisant une relation topologique ou une relation de *direction* de *type carreau unique* (voir équation 7.3), tandis que le deuxième est spécifié par rapport à une relation de *distance* ou par rapport à une relation de *direction* de *type carreaux multiples* (voir équation 7.4).

$$C_{TopDir} = \Big\{ \langle so, spRel \rangle \,|so \in (\texttt{SpatialObject})^C,\ spRel \in V_{OP} : spRel \in \{\texttt{qsr:spatiallyRelated}, \texttt{qsr:North}, \texttt{qsr:West}, \texttt{qsr:South}, \texttt{qsr:East}, \texttt{qsr:Neutral}\}\Big\} \tag{7.3}$$

$$C_{DirDist} = \Big\{ \langle so, spRel \rangle \,|so \in (\texttt{SpatialObject})^C,\ spRel \in (\texttt{qsr:ComposedDirection})^C \cup \cup (\texttt{qsr:DistanceRelation})^C \Big\} \tag{7.4}$$

Nous proposons d'utiliser ONTOAST pour mieux contrôler le processus de construction des $onto-paths$. Concrètement, pour trouver tous les $onto-paths$ qui instancient une séquence de propriétés PS, on tente de construire toutes les combinaisons possibles entre les instances appartenant aux domaines et aux images des expressions à base de propriétés incluses dans PS, tel que l'$onto-path$ résultant satisfasse la définition 7.2 (page 202). Ce processus est décrit en détail dans la section 7.4.2. Chaque fois qu'on analyse une instance o_i dont l'emprise spatiale (exprimée par sa géométrie ou inférée par le système) ne respecte pas l'une des relations spatiales qualitatives $spRel$ par rapport aux objets spatiaux correspondants so, $((so, spRel) \in C_{Spatial})$, la construction de l'$onto-path$ est arrêtée, car toute solution qui inclut o_i n'est pas valide par rapport au contexte spatial. Les instances qui n'ont pas de dimension spatiale sont considérées comme étant cohérentes avec $C_{Spatial}$.

Pour illustrer notre approche, considérons l'ontologie présentée dans la figure 7.6. Lors de l'étude de cette ontologie, un utilisateur impliqué dans la sécurité d'un territoire, pourrait être intéressé par la découverte de liens directs ou indirects entre différents individus : banques, équipes, gens, *etc.*, mais exclusivement dans l'espace relevant de la juridiction de son entreprise de surveillance, ou, plus généralement, dans un territoire ou une région donnés. L'utilisateur peut alors définir un ou plusieurs contextes spatiaux pour sa requête. Deux exemples de tels contextes sont : $C_{TopDir} = \{< France, \texttt{qsr:contains} >\}$[2] et $C_{DirDist} = \{\langle org_1, \texttt{qsr:veryClose} \rangle\}$.

[2] *France* est une instance de la classe `SpatialObject` définie dans l'ontologie de référence.

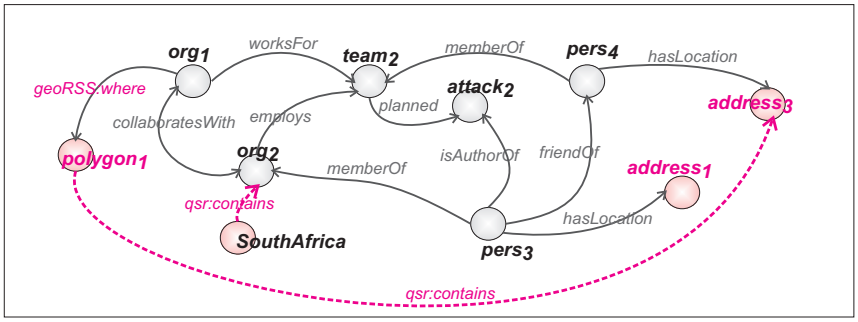


FIG. *7.6: Exemple d'ontologie dans le domaine du terrorisme. Deux types d'instances sont définies : des instances thématiques et des instances spatiales.*

Le premier contexte limite la recherche à l'ensemble des $onto-paths$ pour lesquels tous les individus o_i qui disposent de caractéristiques spatiales ont une emprise spatiale *incluse* dans la région *France*. Le deuxième contexte impose comme filtre spatial la *proximité* par rapport à l'objet org_1. Examinons, par exemple, l'$onto-path$

$$p_1 = org_1 \overset{inverseOf(\texttt{collaboratesWith})}{\rightarrow} org_2 \overset{inverseOf(\texttt{memberOf})}{\rightarrow} pers_3 \overset{\texttt{isAuthorOf}}{\rightarrow} attack_2,$$

qui illustre une connexion possible entre l'organisation org_1 et l'attaque terroriste $attack_2$. Supposons que $polygon_1$ et $address_1$ représentent des lieux (respectivement, une région, et une paire de coordonnées) en *France*. Dans ce cas, lors de la construction de l'$onto-path$, p_1, ONTOAST vérifie d'abord si l'emprise spatiale de l'objet org_1 (ici décrite à l'aide de la propriété-lien `geoRSS:where`) est incluse dans la région *France*. Étant donné que $polygon_1$ est à l'intérieur de la *France*, et que la relation `veryClose` est réflexive, l'individu org_1 est valide par rapport aux contextes $C_{DirDist}$ et C_{TopDir}, donc la construction de l'$onto-path$ peut se poursuivre. L'étude spatiale de la prochaine instance, org_2, est centrée sur la relation `qsr:contains`, qui relie la région $SouthAfrica$ à org_2. Étant donné que org_2 ne peut pas être à la fois incluse dans $SouthAfrica$[3] et dans *France*, les régions $SouthAfrica$ et *France* étant disjointes, org_2 est considéré comme non valide par rapport au contexte C_{TopDir} et la construction de l'$onto-path$ est abandonnée.

7.3.3 Descriptions temporelles prises en charge

Pour associer une emprise temporelle à un individu décrit dans une ontologie OWL ou OWL 2, on dispose de deux alternatives. La plus immédiate est d'utiliser un ou plusieurs attributs désignant une valeur de l'un des types temporels proposés par les deux langages. L'autre possibilité est de faire appel aux concepts et aux relations temporels définis dans l'ontologie OWL-Time (présentée dans la section 3.4.2). Les descriptions réalisées par rapport à cette ontologie sont traitées par le traducteur OWL-ONTOAST, qui les traduit vers des valeurs de types temporels, et vers des relations temporelles de ONTOAST, comme décrit dans la section 6.3.3. En conséquence, le raisonneur ONTOAST peut exploiter ces descriptions temporelles dans des raisonnements. La figure 7.7 rappelle les principales classes temporelles introduites par l'ontologie OWL-Time et définit la relation `td:holds` à travers laquelle on peut attacher à un objet une entité temporelle qui spécifie son emprise temporelle. Les objets temporels étant également des instances de la classe `time:TemporalEntity`, ils peuvent être caractérisés en utilisant toutes les relations temporelles

[3] *France* et *SouthAfrica* sont des instances de la classe `SpatialObject` définies dans l'ontologie de référence, pour lesquelles on connaît la relation topologique : $(France, SouthAfrica) \in (\texttt{qsr :disjoint})^{OP}$.

introduites par l'ontologie OWL-Time.

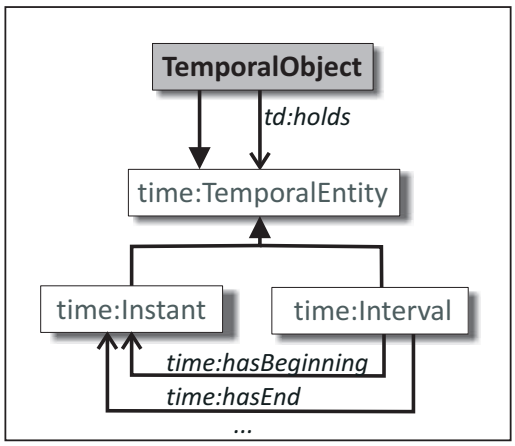


FIG. *7.7: L'ontologie td (acronyme de temporal description) qui rappelle les descriptions temporelles prises en compte par l'analyse sémantique temporelle.*

Concernant l'emprise temporelle des propriétés-lien, à ce jour, il n'est pas possible d'exprimer la validité temporelle des *tuples* dans les frontières décidables d'OWL-DL et de OWL 2 DL. En OWL-Full, les propriétés-lien peuvent être utilisées comme des classes, et donc leurs instances peuvent être décrites par des attributs. Néanmoins, dans cette thèse, nous nous intéressons exclusivement aux sous-langages OWL DL et OWL 2 DL qui sont compatibles avec le système ONTOAST (décrit dans le chapitre 6). Ce dernier impose une séparation stricte entre les domaines d'interprétation des classes, des types de données, des relations et des individus, n'étant donc pas compatibles avec OWL-Full.

Une approche intéressante dans ce sens est proposée dans [120], et vise l'estampillage des triplets RDF par des instants ou des intervalles de temps. Comme la spécification actuelle du langage OWL 2 ne prend pas en charge les estampilles temporelles, nous proposons de les simuler par des annotations spécifiques gérées par le traducteur OWL-ONTOAST. Trois exemples d'annotations temporelles sont illustrés dans la figure 7.8. Les deux premières associent des intervalles temporels à deux tuples, et la dernière estampille le tuple correspondant avec un instant de temps. Comme montré dans l'exemple cité auparavant, il est possible de décrire des intervalles de temps infinis en remplaçant l'instant de début ou l'instant de fin par le symbole "-". Ces estampilles temporelles sont traduites en ONTOAST comme des valeurs de la variable `temporalValidity`, dont le type est `interval`. La variable `temporalValidity` est ajoutée de façon automatique dans la définition des relations dont les tuples sont estampillés avec une validité temporelle. Si l'estampille représente un instant de temps, t, elle est traduite en ONTOAST par un intervalle dont l'instant de début et l'instant de fin sont identiques et égaux à t.

```
ObjectPropertyAssertion(Comment("temporalValidity(2005-10-10T10:30:00, -)") collaboratesWith org1 org2)
ObjectPropertyAssertion(Comment("temporalValidity(2003-01-10, 2007-30-06)") memberOf pers3 org2)
ObjectPropertyAssertion(Comment("temporalValidity(2006-08-10)") sd:hasLocation pers3 polygon1)
```

FIG. *7.8: Exemples d'estampilles temporelles.*

7.3.4 Contexte temporel d'une requête sémantique

Le contexte temporel C_{Temp} agit comme un filtre pour limiter la recherche aux individus et aux tuples qui satisfont une certaine relation de proximité temporelle par rapport à un intervalle de temps donné. Dans le processus d'analyse sémantique, nous considérons deux types de contextes temporels :

- l'un explicite, défini en utilisant un intervalle de temps explicite, spécifié par l'utilisateur, et une relation de topologie temporelle parmi les relations gérées par ONTOAST ;
- l'autre implicite, spécifié par rapport à un objet temporel, instance de la classe `TemporalObject`.

Le contexte implicite est spécifié en utilisant un intervalle $[start, end]$ ou une paire $(date, duration)$, ainsi qu'une relation temporelle topologique qui doit être vérifiée par tous les objets et les tuples inclus dans le $onto-path$, par rapport à l'intervalle donné. Lorsque l'intervalle temporel est décrit par une date de début et une durée, les attributs temporels décrivant les tuples ou les objets considérés seront comparés à l'intervalle $[date, date + dur]$.

Le contexte explicite est spécifié en utilisant un objet temporel, *to*, instance de la classe OWL `TemporalObject` ou de la classe ONTOAST correspondante, et une relation temporelle topologique, *tmpRel*. Pour qu'un $onto-path$ soit considéré valide par rapport à un contexte temporel implicite, tout objet ou tuple inclus par l'$onto-path$ doit satisfaire la relation *tmpRel* avec l'objet *to*.

$$
\begin{aligned}
C_{Temp} \quad = \quad & \Big\{ ([start, end] \vee (date, dur) \vee to,\ tmpRel) \,|\, start, end, date \in (\texttt{time:instant})^{C}, \\
& dur \in (\texttt{time:DurationDescription})^{C}, tmpRel \in \{\texttt{before}, \texttt{after}, \texttt{meets}, \\
& \texttt{metBy}, \texttt{overlaps}, \texttt{overlappedBy}, \texttt{starts}, \texttt{startedBy}, \texttt{during}, \texttt{contains}, \\
& \texttt{finishes}, \texttt{finishedBy}, \texttt{equals}\}\}
\end{aligned} \tag{7.5}
$$

L'intervalle temporel considéré, *par défaut*, commence avec l'origine des temps et s'étend jusqu'à l'instant présent, mais il peut être adapté aux besoins de l'utilisateur. Toutes les relations temporelles qualitatives d'ONTOAST (voir section 6.5.1) peuvent être utilisées. La relation implicite est celle d'inclusion temporelle (`during`). Lors de l'analyse d'une instance *e* (tuple ou objet), candidate à l'intégration dans un $onto-path$, le système compare sa validité, exprimée par ses attributs temporels, au contexte temporel fourni, C_{Temp}. Si *e* n'a pas d'attribut temporel, le système considère qu'elle est atemporelle.

Pour l'exemple illustré par la figure 7.9, considérons la requête $ontoPathAssociated(org_1, pers_4)$ et le contexte temporel $C_{Temp} = \{(int_1, \texttt{during}), ([\texttt{2000-01-01}, \texttt{2006-07-10}], \texttt{overlaps})\}$. Ce contexte fait référence à la période de temps, exprimée par l'intervalle int_1, qui modélise l'emprise temporelle attachée à l'équipe $team_2$, et en même temps à la période de temps explicite $[2000-01-01, 2006-07-10]$.

L'un des $onto-paths$ reliant org_1 à $pers_4$ dans la ABox considérée est
$p_2 = org_1 \overset{inverseOf(worksFor)}{\rightarrow} team_2 \overset{inverseOf(memberOf)}{\rightarrow} pers_4$.
En analysant l'emprise temporelle des objets contenus par celui-ci, on observe que le seul objet disposant de caractéristiques temporelles est $team_2$. L'intervalle int_1 respecte la relation `during` avec lui-même et la relation `overlaps` avec l'intervalle $[2000-01-01, 2006-07-10]$. Cependant, lors de l'analyse des caractéristiques temporelles des tuples, on constate que l'instant temporel attaché au tuple $inverseOf(\texttt{memberOf})(team_2, pers_4)$, même s'il est inclus dans l'intervalle

int_1, ne satisfait pas la relation `overlaps` avec l'intervalle $[2000-01-01, 2006-07-10]$. En conséquence, l'*onto – path* p_2 est invalidé et retiré de la liste des solutions.

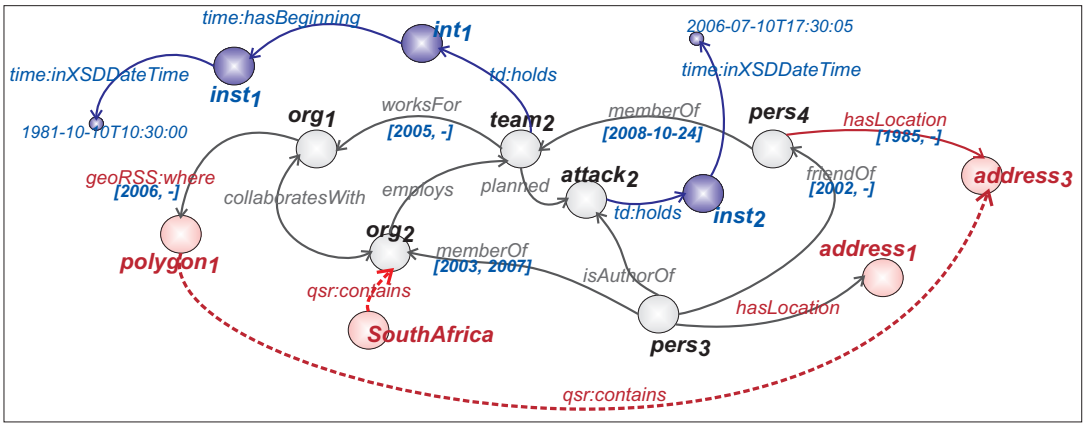


FIG. *7.9: Exemple d'ontologie dans le domaine du terrorisme définissant trois types d'instances : des instances thématiques, des instances spatiales et des instances temporelles. Plusieurs tuples sont étiquetés à l'aide d'intervalles de temps ou d'instants.*

7.4 Framework ONTOAST pour la découverte d'associations sémantiques

7.4.1 Architecture

Le *framework* que nous proposons pour la découverte des associations sémantiques spatio-temporelles entre individus décrits dans des ontologies OWL 2 DL, est illustré dans la figure 7.10. Celui-ci comporte cinq modules principaux, organisés comme des outils d'exploitation distincts construits au-dessus du système ONTOAST. Afin d'exploiter les raisonnements fournis par ONTOAST, les connaissances ontologiques doivent être traduites vers ONTOAST et stockées dans un entrepôt de connaissances orienté-objet (étape 1, figure 7.10). Pour interroger l'entrepôt, l'utilisateur spécifie sa requête (étape 2, figure 7.10) sous la forme d'une paire d'individus (x,y) en utilisant l'*Interface d'interrogation*, qui permet aussi la spécification des *Contextes Spatial* et *Temporel* attachés à la requête. Afin de réduire l'espace de recherche, le processus de découverte des *onto – paths* (étape 3, figure 7.10) débute par le filtrage des connaissances disponibles, en suivant les spécifications des *Contextes Spatial* et *Temporel*. Les connaissances filtrées sont exploitées par le module de *Découverte des onto – paths*. Le Raisonneur ONTOAST est utilisé à la fois dans la phase de filtrage (pour déduire des caractéristiques géographiques des individus, ainsi que la validité temporelle des individus et des tuples), et dans la phase de découverte des *onto – paths* (pour déduire les connexions spatiales entre individus). Les *onto – paths* découverts sont ensuite classés par le module de *Classification de résultats* (étape 4, figure 7.10), en utilisant les spécifications de contexte, et présentés à l'utilisateur par le module de *Visualisation des résultats* (étape 5, figure 7.10).

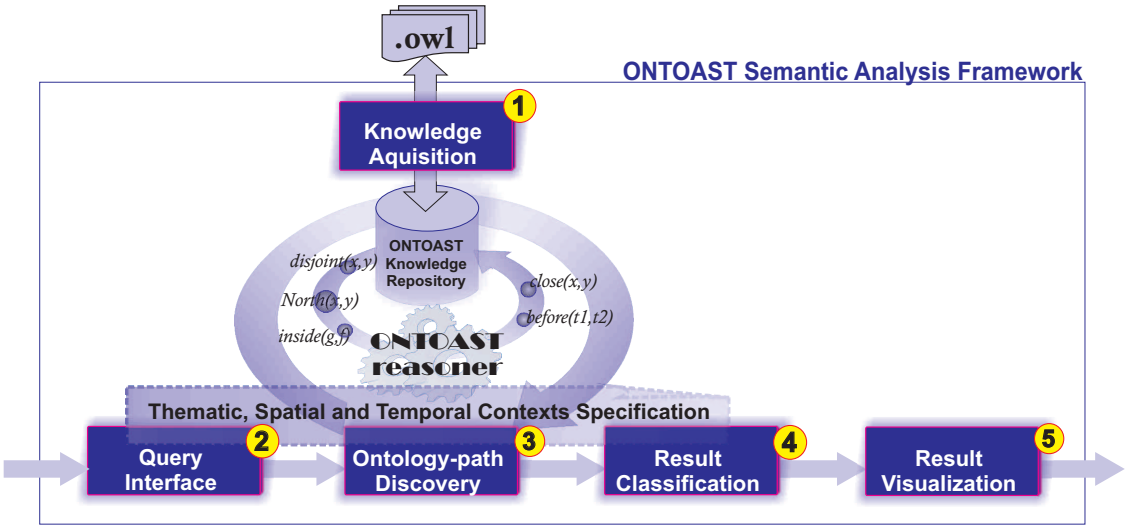


FIG. *7.10: L'architecture du système d'analyse sémantique spatio-temporelle.*

7.4.2 Approche algorithmique pour la découverte d'associations sémantiques

La stratégie que nous avons adoptée pour la découverte d'associations sémantiques reliant deux individus x et y dans une ontologie de référence O, comporte deux étapes progressives. Tout d'abord, nous utilisons les connaissances terminologiques définies dans la fermeture transitive de l'ontologie O pour décider si une association sémantique peut exister entre x et y (voir section 7.4.2.1). Pour cela, nous vérifions qu'au moins une séquence de propriétés $PS_1 = \lfloor OPE_1 \bullet OPE_2 \bullet \ldots \bullet OPE_n \rfloor$ peut être construite de manière à ce que :

$$\exists CE_x, CE_y : x \in (CE_x)^C \wedge CE_x \in domain(OPE_1) y \in (CE_y)^C \wedge CE_y \in range(OPE_n). \quad (7.6)$$

Si une telle séquence de propriétés ne peut pas être construite, la recherche au niveau assertionel n'est pas poursuivie car la requête n'a pas de résultat. Si, au contraire, la construction d'une ou plusieurs séquences de propriétés qui respectent 7.6 est possible, alors, en utilisant une version adaptée d'un algorithme classique de recherche de chemins dans des graphes, comme l'algorithme de recherche en profondeur d'abord [95], on construit toutes les séquences de propriétés reliant les expressions à base de classes auxquelles les individus x et y sont attachés (voir section 7.4.2.2).

L'idée de cette approche, adoptée également par [18, 19, 218], est de réduire la complexité de l'algorithme de recherche des associations sémantiques, en utilisant les connaissances de la TBox. Les dimensions de la TBox étant en général beaucoup moins importantes que celles de la ABox, la recherche au niveau du schéma prend donc *a priori* moins de temps qu'une recherche au niveau des données [218].

Dans un deuxième temps (voir section 7.4.2.3), les séquences de propriétés construites par rapport à la TBox, sont utilisées pour guider la construction des $onto-paths$. Concrètement, celles-ci servent à élaguer la liste des tuples dans lesquels un individu a est impliqué et qui seront exploités pour la construction des $onto-paths$.

7.4.2.1 Construction du graphe terminologique G_{TBox}

Pour toute ontologie OWL 2 DL, O, il est possible de construire un graphe orienté, G_{TBox} défini par l'équation 7.7. L'ensemble de nœuds, N_{TBox}, de G_{TBox} représente l'ensemble d'expressions à

base de classes définies dans la fermeture transitive de l'ontologie O tandis que l'ensemble d'arcs, A_{TBox}, est construit à partir des expressions à base de propriétés-lien définies dans O. La fonction $\lambda : A_{TBox} \rightarrow V_{OPE}$ est une fonction d'étiquetage, qui associe à chaque arc $CE_1 \rightarrow CE_2 \in A_{TBox}$ l'expression à base de propriétés utilisée pour sa définition. Deux nœuds CE_1 et CE_2 sont reliés par un arc $CE_1 \rightarrow CE_2$ si et seulement si, dans l'ontologie O, il existe une expression à base de propriétés OPE telle que son domaine/image inclut l'expression à base de classes CE_1/CE_2 : $CE_1 \in domain(OPE) \ \wedge CE_2 \in range(OPE)$. Les propriétés inverses implicites des propriétés-lien P définies dans O, sont prises en compte afin de saturer le graphe G_{TBox} avec l'ensemble d'arcs implicites. Concrètement, pour chaque propriété-lien P définie dans O, l'ensemble A_{TBox} contient tous les arcs $CE_i \rightarrow CE_j$, où $CE_i \in domain(P)$ et $CE_j \in range(P)$, qui sont étiquetés par $\lambda(CE_i \rightarrow CE_j) = P$, et également les arcs $CE_j \rightarrow CE_i$, qui sont étiquetés par $\lambda(CE_j \rightarrow CE_i) = inverseOf(P)$. Ainsi, construire les séquences de propriétés qui respectent la condition 7.6, revient à trouver les chemins qui relient les nœuds C_x et C_y dans le graphe G_{TBox}.

$$
\begin{aligned}
G_{TBox} &= (N_{TBox}, A_{TBox}, \lambda), \text{où} \\
N_{TBox} &= \{CE | \texttt{l'expression à base de classes } CE \texttt{ est définie} \\
&\quad \texttt{dans la fermeture transitive de O}\} \textit{ et} \\
A_{TBox} &= \{CE_1 \rightarrow CE_2 | \exists OPE \in V_{OPE} : CE_1 \in domain(OPE) \wedge CE_2 \in range(OPE)\} \\
\lambda &: A_{TBox} \rightarrow V_{OPE}, \ \text{où} \ V_{OPE} =_{def} V_{OP} \cup \{\texttt{InverseOf}(P) | P \in V_{OP}\}.
\end{aligned} \tag{7.7}
$$

Afin d'éviter les situations où la ABox contient des $onto-paths$ qui n'ont pas de correspondant au niveau de la TBox (voir l'exemple illustré dans la figure 7.2 section 7.2.2), une phase de pré-calcul doit précéder la construction du graphe G_{TBox}. Celle-ci consiste en la comparaison des *domaines* et des *images* de toute paire de propriétés, (P,Q), définie dans la fermeture transitive de l'ontologie O, en utilisant l'algorithme `bridgeConstruction`. Concrètement, pour chaque paire de propriétés-lien $(P,Q) \in V_{OP} \times V_{OP}$, on vérifie si P ou $\texttt{InverseOf}(P)$ peuvent succéder à Q ou à $\texttt{InverseOf}(Q)$ dans une séquence de propriétés.

Ainsi, l'idée de l'algorithme `bridgeConstruction`, est d'identifier les *ponts* qui existent entre deux expressions à base de propriétés P et Q et qui ne sont pas immédiates en examinant la définition axiomatique de P et de Q. Ces *ponts* existent si, par exemple, une ou plusieurs instances $a \in \Delta_I$ sont attachées à la fois à une des expressions à base de classes incluses dans le *domaine* de P et à une des expressions à base de classes incluses dans l'*image* de Q. Si les instances communes à $domain(P)$ et à $range(Q)$ sont regroupées dans une ou plusieurs expressions à base de classes, CE_{PQ}, qui font partie à la fois du domaine de P et de l'image de Q, alors l'algorithme ne fait rien car le pont entre P et Q existe déjà. Si, au contraire, les instances communes à $domain(P)$ et à $range(Q)$ ne sont pas attachées aux classes communes CE_{PQ}, mais font partie des ensembles D et R (voir lignes 1-2 de la procédure `createBridge`), alors on ajoute à $domain(P)$ et à $range(Q)$ l'expression à base de classes construite comme intersection de D et R (voir lignes 5-6 dans la procédure `createBridge`). Ces ajouts ne modifient pas la sémantique de l'ontologie O, et ils sont éliminés après l'étape de construction des séquences de propriétés.

Notre approche d'identification des *ponts* est différente de celle proposée dans [18, 19], où chaque ressource a définie dans le graphe RDF(S) de référence est analysée afin d'identifier les classes, $C_j, j \geq 1$, auxquelles elle est attachée. Suite à cette analyse, une nouvelle classe d'intersection est ajoutée au modèle : $\sqcap_{i \in 1..j} C_i$, comme sous-classe de chaque classe C_i qui contribue à sa

définition. Ainsi, la classe d'intersection hérite les propriétés qui sont définies pour ses ancêtres, et peut être utilisée comme *pont*. Le désavantage principal de cette approche vient du fait qu'elle s'applique à chaque ressource décrite par le graphe RDF(S), et pas seulement aux éléments du schéma, qui sont moins nombreux. Ainsi, le temps de traitement peut être long si le graphe RDF(S) a des dimensions importantes.

Entrées: $P \in V_{OPE}$, $Q \in V_{OPE}$

Résultat: Le pont entre P et Q, si il existe, est explicite.

1 $D = domain(P) - domain(P) \cap range(Q)$;
2 $R = range(Q) - range(Q) \cap domain(P)$;
3 **si** $(D)^C \cap (R)^C \neq \emptyset$ **alors**
4 $\quad Int = (\sqcup_{C_D \in D} C_D) \sqcap (\sqcup_{C_R \in R} C_R)$;
5 $\quad domain(P) \leftarrow domain(P) \cup \{Int\}$;
6 $\quad range(Q) \leftarrow range(Q) \cup \{Int\}$;
7 **fin**

Procédure `createBridge`(P, Q)

Données: O - l'ontologie de référence

Résultat: Pour chaque $P \in V_{OP}$ le domaine et l'image sont complétés par tous les ponts possibles.

1 **pour tous les** $P, Q \in V_{OP}$ **faire**
2 $\quad createBrigde(domain(P), range(Q))$// P peut être le successeur de Q dans une séquence de propriétés PS
3 $\quad createBrigde(range(P), range(Q))$// $InverseOf(P)$ peut être le successeur de Q dans PS
4 $\quad createBrigde(domain(P), domain(Q))$// P peut être le successeur de $InverseOf(Q)$ dans PS
5 $\quad createBrigde(range(P), range(Q))$// $InverseOf(P)$ peut être le successeur de $InverseOf(Q)$ dans PS
6 **fin**

Procédure `bridgeConstruction`

7.4.2.2 Découverte des séquences de propriétés comme des chemins dans le graphe G_{TBox}

Par la suite, nous présentons l'algorithme que nous utilisons pour la découverte des séquences de propriétés $PS = \lfloor OPE_1 \bullet \ldots \bullet OPE_n \rfloor$, telles que l'expression à base de classes, *start*, fait partie du domaine étendu de OPE_1 et l'expression à base de classes *target*[4] fait partie de l'image étendue de OPE_n. Les différents stades d'évolution de cet algorithme (voir procédure `computePSSet`)

[4]Il est important de faire la différence entre les expressions à base de classes *start* et *target* et les individus $origine(ps)$ et $terminus(ps)$. Si $ps \in (PS)^{OP}$, alors $origine(ps) \in (start)^C$ et $terminus(ps) \in (target)^C$.

ont été rapportés dans [172, 173, 169, 170, 171]. L'ensemble de séquences de propriétés, telles que définies auparavant, est équivalent à l'ensemble de chemins qui relient les nœuds *start* à *target* dans le graphe G_{TBox}. Pour faciliter les traitements ultérieurs, un chemin dans le graphe G_{TBox}, noté GT, représentant une séquence de propriétés OWL 2, notée PS, est décrit par l'intermédiaire d'une séquence de triplets de type $GP = \lfloor (OPE_1, start, CE_1), (OPE_2, CE_1, CE_2), \ldots, (OPE_n, CE_n, target) \rfloor$, où OPE_i est l'expression à base de propriétés de rang i dans PS et CE_{i-1} ($CE_{i-1} \in domain(OPE_i)$) et CE_i ($CE_i \in range(OPE_i)$) sont les expressions à base de classes choisies comme *ponts* entre deux expressions à base de propriétés successives.

La longueur des chemins exploités est limitée par le paramètre *maxLength*. Ainsi, la procédure `computePSSet` peut être vue comme une adaptation de l'algorithme classique de recherche en profondeur d'abord et dont la profondeur est limitée à *maxLength*[5] [9]. La complexité de cet algorithme est, dans le pire des cas, $O(b^{maxLength})$, où b est le nombre maximum d'arcs qui peuvent être attachés à un nœud, et *maxLength* est la profondeur maximale à exploiter. Il est difficile d'estimer les dimensions d'une ontologie (nombre de concepts et de relations) en-dessous desquelles cet algorithme peut fournir des réponses en un temps raisonnable, sans connaître la structure du graphe G_{TBox} (*i.e.* graphe de type complet, aléatoire, grille, clairsemé, petit monde,...) [215]. La raison principale à cela résulte de la difficulté d'estimer le nombre moyen de propriétés définies pour une expression à base de classes. Bien évidemment, on peut considérer que, dans le pire des cas, toutes les expressions à base de propriétés définies dans une ontologie ont comme *domaine* l'ensemble des expressions à base de classes référencées par cette ontologie. Dans ce cas, l'algorithme devient très vite inexploitable. Malheureusement, ce type de limitation est très difficile à éliminer, car le problème de la découverte des chemins entre deux arcs données s et t dans un graphe orienté et avec des cycles, est connu comme étant NP complet.

Néanmoins, l'utilité d'une ontologie qui décrit un ensemble de concepts tous reliés par exactement les mêmes relations est discutable, car le but même d'un modèle est de mettre en évidence les différences qui existent entre les types d'objets qu'il définit. Si, pour chaque paire de classes, on a exactement les mêmes relations il est très difficile, voire impossible de faire la différence entre les classes en question.

Pour réduire encore plus les dimensions du problème, la procédure `computePSSet` exploite seulement une portion du graphe G_{TBox}. Concrètement, les chemins reliant *start* et *target* dans le graphe G_{TBox} sont construits par rapport à un contexte thématique Ct_{Them} (voir la définition dans la section 7.2.5), qui consiste en un ensemble d'expressions à base de classes et d'expressions à base de propriétés, $Ct_{Them} = \{CE_1, \ldots, CE_k, OPE_1, \ldots, OPE_l\}$. L'idée est de limiter la découverte de chemins dans le graphe G_{TBox} à une région thématique qui est considérée comme pertinente pour la requête courante. Cette région thématique est formée de l'ensemble des nœuds qui correspondent aux expressions à base de classes incluses dans Ct_{Them}, auxquels on ajoute l'ensemble des arcs

[5] En anglais "Depth limited search".

dont les étiquettes correspondent aux expressions à base de propriétés incluses par Ct_{Them}.

Données: G_{TBox}

Entrées: *start*, *target*, *currentPS*, *maxLength*, Ct_{Them}, *CtDev*, *currDev*

Sorties: *PSSet* l'ensemble de toutes les séquences de propriétés $PS = \lfloor OPE_1 \bullet \ldots \bullet OPE_n \rfloor$ tel que $n \leq maxLength$ et $start \in domain(OPE_1)$ et $target \in range(OPE_n)$ et qui sont compatibles avec le contexte Ct_{Them}.

```
1  startNotInCt ← false
2  arcNotInCt ← false
3  si sizeOf(currentPS) < maxLength and currDev < CtDev alors
4      si start ∉ Ct_Them alors
5          currDev ← currDev + 1
6          startNotInCt ← true
7      fin
8      sinon
9          currDev ← 0
10     fin
11     pour tous les e ∈ arcsOutOf(start) faire
12         si e ∉ Ct_Them alors
13             currDev ← currDev + 1
14             arcNotInCt ← true
15         fin
16         sinon
17             currDev ← 0
18         fin
19         currentPS.add(e, vertice(start, e))
20         si vertice(start, e) = target alors
21             PSSet ← PSSet ∪ currentPS
22         fin
23         sinon
24             computePSSet (vertice(start, e), target, currentPS, maxLength,
25             Ct_Them, CtDev, currDev)
26             currentPS.remove(e, vertice(start, e))
27             si startNotInCt alors
28                 currDev ← currDev − 1
29             fin
30             si arcNotInCt alors
31                 currDev ← currDev − 1
32             fin
33         fin
34     fin
35 fin
```

Procédure `computePSSet`

Pour qu'un chemin dans le graphe G_{TBox} soit considéré comme valide par rapport au contexte thématique considéré, sa déviation par rapport à Ct_{Them} doit être inférieure au paramètre *CtDev*. En d'autres termes, le chemin ne doit pas contenir plus de *CtDev* éléments successifs qui ne font pas partie du contexte Ct_{Them} :

$GP = \lfloor (OPE_1, start, CE_1), (OPE_2, CE_1, CE_2), \ldots, (OPE_n, CE_{n-1}, target) \rfloor$ est valide par rapport à $Ct_{Them} \Leftrightarrow \nexists k > CtDev : OPE_i, CE_i, OPE_{i+1}, CE_{i+1}, \ldots, OPE_{i+[k/2]} \left[, CE_{i+[k/2]}\right] \notin Ct_{Them}, i \geq 0, i + [k/2] <= n \wedge \nexists k > CtDev : CE_i, OPE_{i+1}, CE_{i+1}, \ldots, OPE_{i+[k/2]} \left[, CE_{i+[k/2]}\right] \notin Ct_{Them}$ (voir lignes 4-6, 12-14 de la procédure `computePSSet`).

Pour mieux gérer les chemins *GP*, nous avons défini la fonction `getTriple(int pos)` qui renvoie le triplet de rang *pos* contenu par *GP* : $(OPE_{pos}, CE_{pos-1}, CE_{pos}), \forall pos \in 1..n$. Pour un triplet de ce type, à l'aide des fonctions `getPropertyExpression()`, `getObject()` et `getSubject()` on obtient respectivement l'expression à base de propriétés OPE_{pos}, l'expression à base de classes CE_{pos-1} et l'expression à base de classes CE_{pos}.

7.4.2.3 Découverte des instances d'une séquence de propriétés

Dans cette section, nous présentons la fonction de découverte des $onto-paths$ reliant deux individus *x* et *y* dans une ontologie de référence, *O*, et qui instancient une séquence de propriétés donnée *GP*. Ces $onto-paths$ sont construits par rapport à un contexte spatial et un contexte temporel donnés.

La procédure `getInstances` parcourt récursivement tous les instances et les tuples attachés aux expressions à base de classes et aux expressions à base de propriétés inclus par *GP*, en essayant de construire des $onto-paths$ qui satisfont le motif donné par *GP*.

À chaque étape *k* dans la construction d'un $onto-path$, la procédure `getInstances` cherche les tuples de l'expression à base de propriétés OPE_k, $OPE_k = GP.getTriple(k).getPropertyExpression()$ qui relient *start* aux instances *o* de l'expression à base de classes $CE_k, CE_k = GP.getTriple(k).getObject()$. L'algorithme prend en compte exclusivement les instances *o* qui sont valides par rapport à Ct_{Spat} et Ct_{Temp}.

Dans le cas où l'expression à base de propriétés OPE_k est l'une des relations spatiales[6] qualitatives définies par l'ontologie *QualitativeSpatialRelations* (où l'inverse d'une relation spatiale), en utilisant le raisonneur ONTOAST, on vérifie si OPE_k est satisfaite par le tuple $\langle start, o \rangle$ (voir lignes 7 à 9). Si c'est le cas, on ajoute les éléments *r* et *o* à l'$onto-path$ courant et on avance dans la récursivité (voir lignes 11 et 12).

Il faut préciser que la relation spatiale obtenue par ONTOAST peut être une disjonction de relations spatiales, dans le cas où, suite à l'utilisation du raisonnement qualitatif, le domaine de la variable représentant la relation spatiale entre *start* et *o* n'a pas pu être réduit à une seule relation (voir section 6.4.2). Il est donc nécessaire de vérifier si OPE_k fait partie de *r* (voir ligne 10).

Dans le cas où OPE_k n'est pas une relation spatiale qualitative, on vérifie si le tuple $\langle start^*, o \rangle$ fait partie de son extension et s'il est valide par rapport au contexte Ct_{Temp} (voir ligne 16). Si c'est le cas, on ajoute a l'$onto-path$ courant la relation OPE_k et l'objet *o* (voir ligne 17) et on continue la construction du *path* en passant à l'étape $k+1$ où le nouveau *start* est *o*. Il est important de souligner que, pour la construction des $onto-paths$ on prend en compte les axiomes d'égalité entre individus, lors de l'étude des tuples dans lesquels l'objet *start* est impliqué (voir ligne 16).

[6] Le même raisonnement s'appliquant pour les relations temporelles, nous avons choisi de ne pas l'illustrer ici, pour des raisons de concision de la présentation.

Ainsi, par $start^*$ on désigne l'individu *start* ainsi que tous les individus qui sont définis comme étant égaux à *start* :

$$start^* =_{def} \{start\} \cup \{x | \langle start, x\rangle \in (\texttt{AROMSameIndividual})^C \vee \langle x, start\rangle \in (\texttt{AROMSameIndividual})^C\}.$$

Données: *BC*

Entrées: $start, target \in \Delta_I$, *path* est l'$onto-path$ courant, *GP* est la séquence de propriétés pour laquelle on cherche les instances, *k* est l'élément de la *GP* en cours de traitement, Ct_{Spat} est le contexte spatial, Ct_{Temp} est le contexte temporel.

Sorties: $res = \left\{p | p \in (GP)^{OP} : origine(p) = start \wedge terminus(p) = target\right\}$.

```
1  si k < GP.size() alors
2      si start = target alors
3          res ← res ∪ {path}
4      fin
5      sinon
6          pour tous les o ∈ (GP.getTriple(k).getObject())^C tels que
           inContext(o, Ct_Spat, CtTemp) faire
7              si isSpatialRelation(GP.getTriple(k).getPropertyExpression()) alors
8                  relType = typeOf(GP.getTriple(k).getPropertyExpression())
                       // relType ∈ {Topology, Distance, Direction}
9                  r ← ONTOAST.getSpatialRelation(start, o, relType)
10                 si r ≠ ∅ et GP.getTriple(k).getPropertyExpression() ∈ r alors
11                     path ← path.add(r, o)
12                     getInstances(o, target, path, GP, k+1, CtSpat, CtTemp)
13                     path ← path.remove(r, o)
14                 fin
15             fin
16             sinon si ⟨start*, o⟩ ∈ (GP.getTriple(k).getPropertyExpression())^OP et
               inContext(GP.getTriple(k), Ct_Temp) alors
17                 path ← path.add(GP.getTriple(k).getPropertyExpression(), o)
18                 getInstances(o, target, path, GP, k+1, Ct_Spat, Ct_Temp)
19                 path ← path.remove(GP.getTriple(k).getPropertyExpression(), o)
20             fin
21         fin
22     fin
23 fin
```

Procédure `getInstances`

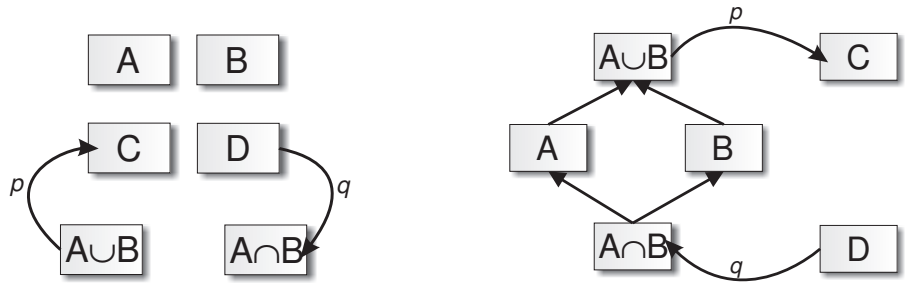
Si l'avancement n'est plus possible, en revenant en arrière, on élimine la dernière paire (OPE_k, *o*) ajoutée à *path*, et on en choisit une autre pour laquelle on applique le même raisonnement.

Lorsque *start* et *target* sont égaux, on a trouvé un $onto-path$ qu'on ajoute à l'ensemble des

solutions (ligne 3).

7.4.2.4 Découverte des associations sémantiques de type *ontoPathAssociated*

Pour trouver tous les *onto* – *paths* qui relient deux individus *x* et *y* dans une ontologie *O*, nous avons défini l'algorithme *onto* – *path*`Discovery`. Celui-ci utilise le raisonneur FaCT++ [230] pour classer l'ontologie de référence *O* (voir ligne 1). Le résultat de la classification est une hiérarchie de classes dans laquelle tous les liens de spécialisation sont explicites pour chaque expression à base de classes. Par exemple, pour l'ontologie illustrée dans la figure 7.11a, la hiérarchie de spécialisation après classification est montrée dans la figure 7.11b.



(a) Hiérarchie de classes avant la classification.

(b) Hiérarchie de classes après la classification.

FIG. *7.11: Résultat de la classification.*

Suite à la classification de l'ontologie, nous construisons le domaine et l'image étendus de chaque expression à base de propriétés (voir ligne 2 dans l'algorithme *onto* – *path*`Discovery`). Le domaine/image étendu(e) d'une expression à base de propriétés, *P*, est construit(e) à partir du domaine/image de *P*, auquel/à laquelle on ajoute toutes les expressions à base de classes qui sont subsumées par $domain(P)/range(P)$. Par exemple, si l'ontologie *O* contient une propriété *P* pour laquelle le domaine est $domain(P) = \{A \cup B\}$ et l'image est $range(P) = \{C\}$ (voir figure 7.11a), après extension on obtient : $domain(P) = \{A \cup B, A, B, A \cap B\}$ ($range(P)$ reste le même). Cette étape d'extension est utile pour la construction des séquences de propriétés, lorsqu'il faut déterminer, par exemple, quelles sont les expressions à base de propriétés pour lesquelles une expression à base de classes donnée, *CE*, est définie comme faisant partie du domaine.

L'algorithme *onto* – *path*`Discovery` continue par la construction des *ponts* entre propriétés (voir ligne 3), à l'aide de la procédure `bridgeConstruction` présentée auparavant. À partir de l'ontologie obtenue, on construit le graphe G_{TBox} (voir ligne 4). La construction de l'ensemble de séquences de propriétés qui existent entre les expressions à base de classe CE_s auxquelles l'individu *s* est attaché, et les expressions à base de classes CE_t pour lesquelles l'individu *t* est une instance, est décrite par les lignes 6- 11. Concrètement, pour chaque paire CE_s, CE_t on calcule à l'aide de la procédure `computePS` l'ensemble exhaustif des chemins qui relient les nœuds CE_s, CE_t dans le graphe G_{TBox}.

Si l'ensemble de séquences de propriétés *PS* est vide, la recherche n'est pas poursuivie car il est impossible de trouver une solution (voir lignes 14-15). Dans le cas contraire, l'ontologie *O* est traduite dans ONTOAST (ligne 18), et, en utilisant un algorithme de recherche en profondeur d'abord similaire à celui utilisé pour la découverte des séquences de propriétés (voir procédure `getInstances`), on construit l'ensemble d'*onto* – *paths* qui correspondent aux individus *s* et *t*.

Notre idée est d'utiliser le raisonneur ONTOAST pour vérifier la compatibilité des caractéristiques spatiales et temporelles des individus et des tuples inclus par chaque $onto-path$ construit, par rapport aux contextes spatial et temporel.

Entrées: $s,t \in \Delta_I$, O, $maxLength$, Ct_{Them}, Ct_{Temp}, Ct_{Sp}, $CtDev$

Sorties: $OntoPath$ l'ensemble des $onto-path$ qui relient s et t dans l'ontologie O, et qui respectent les restrictions contextuelles.

$O \leftarrow FaCT.classify(O)$
$O \leftarrow implicitDomainRange(O)$
$O \leftarrow bridgeConstruction(O)$
$G_{TBox} \leftarrow constructGraph(O)$
$PS \leftarrow \emptyset$
pour tous les $CE_s \in O : s \in (CE_s)^C$ **faire**
  pour tous les $CE_t \in O : t \in (CE_t)^C$ **faire**
    $PSSet \leftarrow \emptyset$
    $currentPS \leftarrow s$
    $computePSSet(CE_s, CE_t, currentPS, maxLength, Ct_{Them}, CtDev, 0)$
    $PS \leftarrow PS \cup PSSet$
  fin
fin
si $PS = \emptyset$ **alors**
  return $\emptyset$
fin
sinon
  $BC \leftarrow owl2Ontoast(O)$
  $OntoPath \leftarrow \emptyset$
  pour tous les $P \in PS$ **faire**
    $OntoPath \leftarrow OntoPath \cup getInstances(s,t,\emptyset,P,1,Ct_{Spat},Ct_{Temp})$
  fin
  $update(O,BC)$
  $eliminateBridges(O)$
  return $OntoPath$
fin

Algorithme 8: $onto-path$Discovery

7.4.3 Étude de cas

Les capacités de raisonnement spatial d'ONTOAST s'avèrent très utiles pour l'analyse sémantique, car elles permettent de construire des $onto-paths$ entre des individus, sur la base d'une proximité spatiale et/ou temporelle intéressante. Par la suite, nous illustrons exclusivement l'approche d'utilisation des relations spatiales lors de la construction des $onto-paths$. L'utilisation

des relations temporelles étant très similaire, nous ne la traitons pas ici.

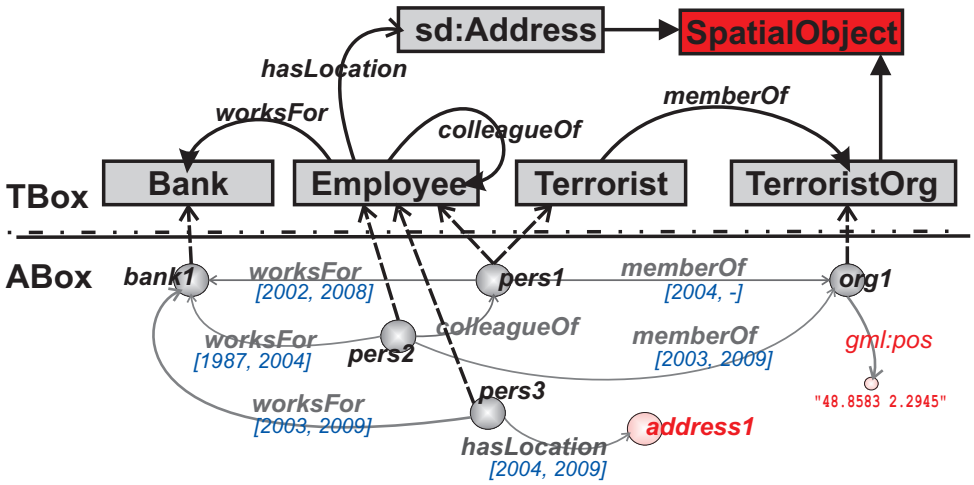


FIG. *7.12: Extrait d'une ontologie qui modélise le domaine de la sécurité d'un territoire.*

Considérons l'exemple de l'ontologie O illustré dans la figure 7.12 et la requête $ontoPathAssociated(bank_1, org_1)$, avec une limite de longueur fixée à 3 : $maxLength = 3$. Nous fixons le contexte thématique pour cette requête à $Ct_{Them} = \{$`Bank`, `TerroristOrg`, `Address`, `worksFor`, `qsr:North`, `qsr:veryClose`$\}$ et la déviation contextuelle à $CtDev = 2$. Nous considérons également le contexte temporel $Ct_{Temp} = \{[2005, 2009]$, `overlaps`$\}$.

Étant donné que les classes `Address` et `TerroristOrg` sont définies comme sous-classes de la classe `SpatialObject`, on peut attacher une description spatiale à leurs instances, en utilisant les propriétés définies pour les objets spatiaux par l'ontologie *GeoRSS-Simple*. De même, les relations spatiales qualitatives définies dans l'ontologie *QualitativeSpatialRelations* pour des objets spatiaux peuvent également être utilisées pour décrire les relations spatiales qui existent entre leurs instances.

La première étape d'analyse consiste en la construction des domaines et des images étendues (voir section 7.4.2.4) pour les propriétés-lien définies dans O ainsi que pour les relations spatiales définies dans l'ontologie *QualitativeSpatialRelations* et considérées intéressantes pour la requête courante (`qsr:North`, `qsr:subject` et `qsr:object`[7]). On obtient :
$domain($`worksFor`$) = \{$`Employee`$\}$; $range($`worksFor`$) = \{$`Bank`$\}$
$domain($`colleagueOf`$) = \{$`Employee`$\}$; $range($`colleagueOf`$) = \{$`Employee`$\}$
$domain($`memberOf`$) = \{$`Terrorist`$\}$; $range($`memberOf`$) = \{$`TerroristOrg`$\}$
$domain($`hasLocation`$) = \{$`Employee`$\}$; $range($`hasLocation`$) = \{$`Address`$\}$
$domain($`qsr:North`$) = \{$`Address`, `TerroristOrg`$\}$
$range($`qsr:North`$) = \{$`Address`, `TerroristOrg`$\}$
$domain($`qsr:veryClose`$) = \{$`Address`, `TerroristOrg`$\}$
$range($`qsr:veryClose`$) = \{$`Address`, `TerroristOrg`$\}$.

L'algorithme continue par la construction des ponts entre les propriétés-lien, comme décrit dans la section 7.4.2.1. Ceci a pour effet l'ajout de l'expression à base de classes `Employee` $\cap$ `Terrorist` au domaine et à l'image de la propriété `colleagueOf` ainsi qu'aux domaines des

[7] Les relations `qsr :subject` et `qsr :object` sont prises en compte car elles participent à la définition de la relation `veryClose`.

propriétés `worksFor`, `memberOf` et `hasLocation` :
domain(`colleagueOf`) = {`Employee`, `Employee`∩`Terrorist`};
range(`colleagueOf`) = {`Employee`, `Employee`∩`Terrorist`};
domain(`memberOf`) = {`Employee`, `Employee`∩`Terrorist`};
domain(`worksFor`) = {`Employee`, `Employee`∩`Terrorist`};
domain(`hasLocation`) = {`Employee`, `Employee`∩`Terrorist`};

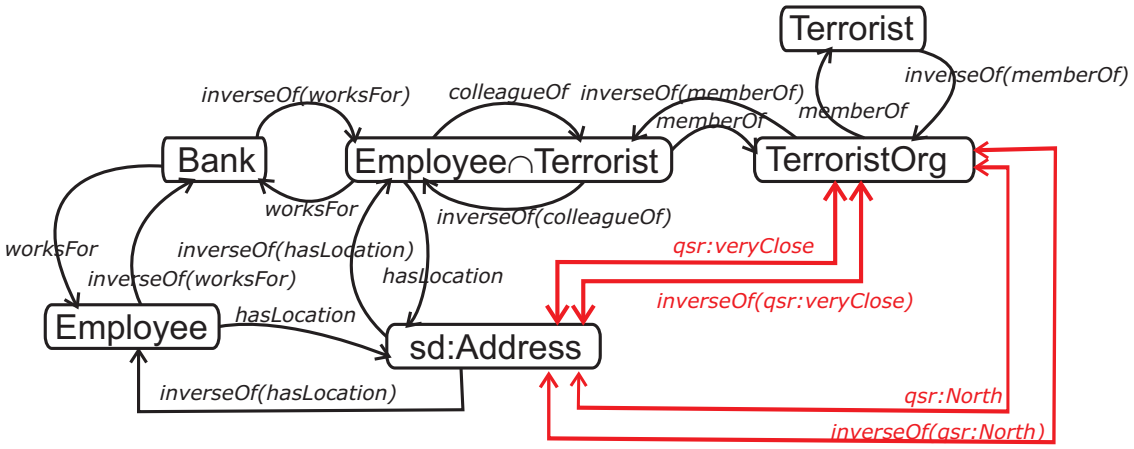


FIG. *7.13: Le G_{TBox} construit à partir de l'ontologie illustrée sur la figure 7.12. Les flèches à double sens reliant les nœuds,* `Address` *et* `TerroristOrg`*, sont utilisées pour rendre la figure plus lisible. Elles remplacent les paires d'arc* `Address`→ `TerroristOrg` *et* `TerroristOrg`→ `Address`*, dont les étiquettes sont identiques (i.e.*`qsr :North` *et* `inverseOf(qsr :North)` *respectivement).*

L'étape de construction du graphe G_{TBox} produit le graphe illustré sur la figure 7.13. Celui-ci est obtenu à partir de l'ontologie O, saturée avec toutes les expressions à base de propriétés implicites (voir section 7.4.2.1). À partir de ce graphe, en utilisant la procédure `computePSSet`, on trouve tous les chemins qui relient les classes `Bank` et `TerroristOrg`. Parmi les séquences de propriétés obtenues on retrouve :

$GP_1 = \lfloor$(*inverseOf*(`worksFor`), `Bank`, `Employee`∩`Terrorist`),
(*memberOf*, `Employee`∩`Terrorist`, `TerroristOrg`)$\rfloor$
$GP_2 = \lfloor$(*inverseOf*(`worksFor`), `Bank`, `Employee`∩`Terrorist`),
(`hasLocation`, `Employee`∩`Terrorist`, `Address`),
(`qsr:North`, `Address`, `TerroristOrg`)$\rfloor$
$GP_3 = \lfloor$(*inverseOf*(`worksFor`), `Bank`, `Employee`∩`Terrorist`),
(`hasLocation`, `Employee`∩`Terrorist`, `Address`),
(`qsr:veryClose`, `Address`, `TerroristOrg`)$\rfloor$[8]
$GP_4 = \lfloor$(*inverseOf*(`worksFor`), `Bank`, `Employee`),
(`hasLocation`, `Employee`, `Address`),
(`qsr:North`, `Address`, `TerroristOrg`)$\rfloor$
$GP_5 = \lfloor$(*inverseOf*(`worksFor`), `Bank`, `Employee`),
(`hasLocation`, `Employee`, `Address`),
(`qsr:veryClose`, `Address`, `TerroristOrg`)$\rfloor$

[8]La longueur des séquences GP_2 et GP_3 est 3 car les 2 triplets utilisés pour leur définitions (*i.e.* `subject(veryClose1, Address)` et `object(veryClose1, TerroristOrg)`) sont considérés comme un seul triplet.

...

Une autre séquence de propriétés reliant les deux classes est :

$GP_6 = \lfloor (inverseOf(\texttt{worksFor}), \texttt{Bank}, \texttt{Employee} \cap \texttt{Terrorist}),$
$(colleagueOf, \texttt{Employee} \cap \texttt{Terrorist}, \texttt{Employee} \cap \texttt{Terrorist}),$
$(memberOf, \texttt{Employee} \cap \texttt{Terrorist}, \texttt{TerroristOrg}) \rfloor.$

Celle-ci n'est cependant pas valide dans le contexte thématique courant, car elle contient une séquence de trois éléments (`Employee` $\cap$ `Terrorist`, `colleagueOf`, `Employee` $\cap$ `Terrorist`), qui ne font pas partie du Ct_{Them}, alors que la déviation contextuelle acceptée est 2.

Pour les séquences de propriétés découvertes GP_1, $GP_2, \ldots$, on cherche les instances à l'aide de l'algorithme `getInstances`. Dans le cas de la séquence GP_1, on trouve comme seule instance valide l'$onto-path$:

$p_1 = bank_1 \overset{inverseOf(\texttt{worksFor})}{\rightarrow} pers_1 \overset{\texttt{memberOf}}{\rightarrow} org_1$

Une autre instance de GP_1 est :

$p_2 = bank_1 \overset{inverseOf(\texttt{worksFor})}{\rightarrow} pers_2 \overset{\texttt{memberOf}}{\rightarrow} org_1.$

Cependant la construction de celle-ci ne peut pas aboutir en raison de l'incompatibilité entre la validité temporelle du tuple $\texttt{worksFor}(pers_2, bank_1)$ et le contexte Ct_{Temp}.

Supposant que les relations spatiales déduites par ONTOAST entre l'adresse $address_1$ et le point 48.8583, 2.2945 sont `veryClose` et `North`, deux nouveaux $onto-paths$ sont découverts, instances des séquences de propriétés GP_4 et GP_5 :

$p_3 = bank_1 \overset{inverseOf(worksFor)}{\rightarrow} pers_3 \overset{hasLocation}{\rightarrow} address_1 \overset{North}{\rightarrow} org_1.$

$p_4 = bank_1 \overset{inverseOf(worksFor)}{\rightarrow} pers_3 \overset{hasLocation}{\rightarrow} address_1 \overset{veryClose}{\rightarrow} org_1.$

En d'autres termes, la personne $pers_3$ qui travaille pour la banque $bank_1$ habite très près de l'organisation org_1, et au nord par rapport à celle-ci. Une association sémantique est utile pour donner des indices par rapport aux relations qui peuvent exister entre différentes individus. Toute latitude est donnée à l'utilisateur pour juger de l'utilité et l'importance de ces relations. Ainsi, le fait que la personne $pers_3$ habite très près de l'organisation org_1, n'implique pas forcément l'appartenance de la personne à l'organisation, mais souligne une telle possibilité, suggérée par la proximité spatiale existant entre ces deux objets.

Ce type d'association sémantique, impliquant des relations spatiales qualitatives calculées/ inférées à la volée, ne peut pas être découvert en utilisant les approches d'analyse sémantique existantes à ce jour.

7.5 Synthèse

Dans ce chapitre, nous avons proposé une adaptation de l'analyse sémantique pour les ontologies OWL 2 DL. Celle-ci exploite les axiomes offertes par OWL 2 afin d'expliciter les informations implicites, qui sont ensuite utilisées pour la déduction de liens complexes entre individus, dans le cadre du Web Sémantique.

Nous avons également illustré ici, l'utilisation d'ONTOAST dans le processus de découverte d'associations sémantiques. Nous avons décrit l'architecture du système d'analyse sémantique spatiale et temporelle que nous proposons, et qui utilise les capacités de raisonnement spatial et temporel d'ONTOAST, d'une part pour limiter l'espace de recherche, et, d'autre part, pour construire de nouvelles associations sémantiques.

Dans le premier cas, ONTOAST filtre les connaissances ontologiques pour ne retenir que celles qui sont compatibles avec le contexte spatial et temporel spécifié par l'utilisateur. Dans le deuxième cas, en utilisant les informations spatiales et temporelles décrivant des individus et des tuples, ONTOAST peut déduire des relations spatiales implicites. Ces relations seront ensuite utilisées pour la construction de nouvelles associations sémantiques.

Bilan et perspectives

Chapitre 8

Bilan et perspectives

8.1 Bilan

L'objectif de cette thèse est d'étudier les défis mais surtout les avantages qui découlent d'une exploitation adaptée des informations spatiales et temporelles décrites dans le cadre du Web Sémantique Géospatial. Cette étude est centrée sur les techniques de raisonnement à base d'information spatiale et temporelle qui peuvent être utilisées pour exploiter des descriptions OWL DL et OWL 2 DL et sur leur application dans le domaine de l'*analyse sémantique*.

Ainsi, notre but est de proposer un *framework* pour la découverte d'*associations sémantiques* entre individus, capable d'exploiter de façon adaptée les informations spatiales et temporelles attachées aux individus, d'une part pour limiter l'espace de recherche à une région de l'espace et/ou à une période de temps données, et, d'autre part, pour construire de nouvelles associations sémantiques, sur la base d'une proximité spatiale et temporelle intéressante qui existe entre individus.

Dans ce chapitre, nous présentons le bilan de notre proposition, ainsi que quelques perspectives de nos travaux. Notre proposition est organisée selon trois axes :

a) la construction de AROM-ONTO, un système de Représentation de Connaissances par Objets, compatible avec le langage OWL 2 DL ;

b) l'intégration, au système AROM-ONTO, d'un module de représentation et de raisonnement spatial et temporel, appelé ONTOAST ;

c) la construction d'un *framework* pour la découverte d'associations sémantiques pour les ontologies OWL 2 DL, utilisant le raisonneur ONTOAST pour exploiter les informations spatiales et temporelles attachées aux individus.

a) AROM-ONTO

L'extension AROM-ONTO est issue d'une étude de positionnement du système de Représentation des Connaissances par des Objets AROM par rapport au langage OWL 2 DL, et du constat de la complémentarité des systèmes de typage et des raisonnements qu'ils offrent. Souhaitant tirer profit de cette complémentarité, nous avons réalisé un travail d'extension du méta-modèle d'AROM afin que celui-ci intègre un ensemble maximal de constructions validées par OWL 2 DL. Le nouveau méta-modèle permet, par exemple, la définition de classes à partir d'opérations ensemblistes ou d'énumération, la définition d'axiomes, la définition de caractéristiques pour les propriétés (des associations binaires ayant une structure prédéfinie)... L'objectif de ce travail est de garantir un minimum de perte d'informations lors de l'importation/l'exportation d'une ontologie dans/depuis AROM-ONTO.

Le système obtenu, AROM-ONTO, permet, par le biais d'un moteur de traductions à base de règles XSLT, l'importation d'ontologies OWL 2 DL vers AROM et l'exportation vers le Web Sémantique, via OWL 2 DL, de bases de connaissances. D'une part, l'objectif est que les capacités

de vérification et d'inférence d'AROM-ONTO soient sollicitées sur de ontologies importées du Web Sémantique, afin de combler les lacunes des outils de raisonnement dédiés à OWL/OWL 2. D'autre part, AROM-ONTO peut être mis à profit en tant qu'outil d'édition et de complétion d'ontologies, soit de manière graphique à l'image de Protégé, soit par programmation via l'API, avant que ces bases de connaissances ne soient mises à disposition du Web Sémantique sous forme d'ontologies. Enfin, la compatibilité entre AROM-ONTO et OWL 2 rend possible l'utilisation, via des appels distants, de raisonneurs conçus pour OWL/OWL 2, augmentant ainsi les possibilités d'inférence pour AROM-ONTO lui-même.

b) ONTOAST

ONTOAST est une extension d'AROM-ONTO qui offre une gestion et une exploitation adaptées des informations spatiales et temporelles quantitatives, ainsi que des relations spatiales et temporelles qualitatives. Il propose plusieurs algorithmes de raisonnement spatial et temporel, et peut être utilisé dans le contexte du Web Sémantique Géospatial pour exploiter les annotations spatiales et temporelles décrites en OWL DL ou OWL 2 DL. Il utilise les types temporels proposés par les deux langages et/ou les concepts spatiaux et temporels définis par les ontologies *GeoRSS-Simple*, *OWL-Time* et *QualitativeSpatialRelations*. Enfin, l'ontologie *QualitativeSpatialRelations* est une ontologie que nous proposons pour la représentation des relations spatiales qualitatives compatibles avec les relations spatiales utilisées par ONTOAST.

Le raisonnement quantitatif proposé par ONTOAST est basé sur l'exploitation d'un ensemble d'opérateurs spatiaux que nous avons ajoutés au module spatio-temporel AROM-ST. Le raisonnement qualitatif, quant à lui, est basé sur les définitions axiomatiques des relations spatiales/ temporelles proposés par les approches de représentation qualitative de l'espace/du temps étudiées dans l'état de l'art. L'exploitation des relations qualitatives et de leurs définitions axiomatiques est réalisée à travers l'utilisation d'un réseau de contraintes.

Les relations spatiales et temporelles qui sont explicitées à l'aide de ONTOAST peuvent être exportées vers le Web Sémantique afin d'être mises à la disposition des raisonneurs et utilisées pour répondre à des requêtes.

Le système ONTOAST a été utilisé dans le cadre du projet PhotoMap [233] pour l'annotation automatique de documents multimédias créés en des situation de mobilité. Concrètement, un utilisateur mobile, disposant d'un dispositif mobile doté d'une caméra et de différents capteurs (*i.e.* GPS, senseur de température, altimètre *etc.*), produit des documents multimédias (*i.e.* photos, vidéos, *etc.*) qui sont automatiquement annotés par rapport à une ontologie de contexte, appelée *ContextMultimedia* [232]. Celle-ci permet, entre autres, la description des informations spatiales et temporelles quantitatives et qualitatives. En d'autres mots, l'ontologie *ContextMultimedia* fait appel aux concepts et aux relations définis dans les ontologies *GeoRSS-Simple* (voir section 3.3.1, page 72), *QualitativeSpatialRelations* (que nous proposons dans la section 6.18, page 185) et *OWL-Time* (voir section 3.4.2, page 78).

Chaque document multimédia est annoté avec des données quantitatives (*i.e.* la position GPS où le document a été créé, l'heure et la date de la création, *etc.*), obtenues à partir de capteurs, mais aussi avec des relations spatiales et temporelles qualitatives inférées en utilisant le raisonneur ONTOAST. Pour ne donner qu'un exemple, la figure 6.15 (voir page 171) illustre une annotation spatiale, où `currentPos` est la position GPS qui correspond à l'endroit où un document

multimédia a été créé. Ensuite, en utilisant le service Web "*Find nearby Wikipedia Entries*"[1], un système d'enrichissement de contexte identifie les dix points d'intérêt les plus proches par rapport à `currentPos`. Entre `currentPos` et chaque point d'intérêt identifié auparavant, le système ONTOAST calcule les relations topologique, de direction et de distance. Dans notre exemple, l'une de relations calculées par ONTOAST est `veryClose` entre `currentPos` et la tour Eiffel. Les relations obtenues sont ajoutées dans la ABox de l'ontologie *ContextMultimedia* et seront utilisées pour répondre à des requêtes telles que :

- quelles sont les photos que j'ai prises au *Nord* de la tour Eiffel ?
- quelles sont les photos prises à l'*intérieur* du Stade de France ?
- quelles sont les vidéos filmées à proximité de la tour Eiffel ? ...

Dans le cadre de cette thèse, nous avons utilisé les capacités de raisonnement offertes par ONTOAST pour la découverte d'*associations sémantiques* entre individus décrits dans des ontologies OWL 2 DL.

c) Analyse sémantique pour les ontologies OWL 2 DL

Partant du constat des limites de l'analyse sémantique dans le cadre des graphes RDF(S), nous avons proposé son adaptation pour les ontologies OWL 2 DL. Celle-ci exploite les axiomes offerts par OWL 2 afin de rendre explicites certaines informations implicites, qui sont ensuite utilisées pour la déduction de liens complexes qui existent entre individus, dans le cadre du Web Sémantique.

Nous tirons avantage ici d'ONTOAST dans le processus de découverte d'associations sémantiques. Nous avons décrit l'architecture du système d'analyse sémantique spatiale et temporelle que nous proposons, et qui utilise les capacités de raisonnement spatial et temporel d'ONTOAST pour, d'une part limiter l'espace de recherche, et, d'autre part construire de nouvelles associations sémantiques. Dans le premier cas, ONTOAST filtre les connaissances ontologiques pour ne finalement retenir que celles qui sont compatibles avec le contexte spatial et temporel spécifié par l'utilisateur. Dans le deuxième cas, en utilisant les informations spatiales et temporelles décrivant des individus et des tuples, ONTOAST peut déduire des relations spatiales implicites. Ces relations sont ensuite exploitées pour la construction de nouvelles associations sémantiques.

8.2 Perspectives

L'étude que nous avons présentée dans cette thèse nous laisse envisager des perspectives intéressantes pour la suite de nos travaux. Nous classons ces futurs travaux en trois catégories : à court terme, à moyen terme et à long terme.

8.2.1 Perspectives à court terme

Le travail décrit dans cette thèse étant principalement conceptuel, nous envisageons comme perspective à court terme, l'implémentation d'un prototype pour le système AROM-ONTO. Aussi, nous souhaitons étudier l'impact que les nouvelles structures de représentation définies pour AROM-ONTO ont sur l'algorithme de classification d'instances proposée par AROM. Il s'agit notam-

[1] http ://www.geonames.org/export/wikipedia-webservice.html

ment d'étudier les effets de la prise en charge des nouveaux axiomes qui définissent des individus comme étant égaux et les classes, les propriétés et les variables comme étant équivalentes.

À l'instar de Protégé, ce prototype AROM-ONTO devra être couplé avec une base de données afin de permettre l'exploitation des ontologies de dimensions étendues. Dans cette direction, nous souhaitons également tester les performances de la classification des instances de classes et de propriétés-lien comme mécanisme de raisonnement pour des ontologies OWL/OWL 2.

Du point de vue applicatif, nous poursuivons le développement de AROM-ONTO avec l'implémentation de l'extension ONTOAST. Nous envisageons la mise en place d'un ensemble de tests pour étudier le passage à l'échelle des algorithmes proposés.

En tant qu'objectif principal pour l'analyse sémantique, nous souhaitons mettre en œuvre un prototype du *framework* d'analyse sémantique spatio-temporelle que nous proposons dans le cadre de cette thèse et tester nos algorithmes sur une base ontologique large, afin de quantifier leur performance dans des scénarios réels de découverte d'associations sémantiques.

8.2.2 Perspectives à moyen terme

Comme perspective à moyen terme, nous envisageons une étude encore plus poussée sur les apports du couplage entre un SRCO comme AROM et un langage à ontologies comme OWL. Dans cette optique, il est intéressant de juger dans quelles conditions l'introduction en OWL et OWL 2 de définitions d'attributs via des équations (à l'image des équations LMA proposées par AROM) est utile. Ces équations pourraient être définies en utilisant des propriétés d'annotations qui respectent une syntaxe donnée, comprise par le traducteur $AROM - ONTO \leftrightarrow OWL2$. Ignorées par les raisonneurs construits autour de OWL, ces équations ne changent pas la sémantique de l'ontologie. L'avantage de l'approche est que ces équations pourraient être exploitées par AROM-ONTO, moyennant une adaptation du traducteur $AROM - ONTO \leftrightarrow OWL2$, pour compléter les valeurs manquantes des attributs. Suite à ce processus de saturation de la base ontologique avec toutes les valeurs implicites des attributs, il faut néanmoins vérifier que l'ontologie résultat reste cohérente.

Dans l'environnement ouvert et collaboratif du Web Sémantique, de nombreuses ontologies peuvent être utilisées pour décrire les mêmes individus. Leur contenu est souvent issu de différentes sources d'informations (*i.e.* utilisateurs, flux de nouvelles, *etc.*) ou inféré à l'aide de raisonneurs. Ces ontologies peuvent donc contenir des incohérences causées par des assertions contradictoires. Par exemple, dans le cas d'une ontologie spatio-temporelle, une incohérence peut être causée par la définition des relations $\texttt{inside}(A,B)$, $\texttt{inside}(A,C)$ et $\texttt{disjoint}(B,C)$. Dans ce cas, on souhaite disposer d'une méthode qui nous permette d'éliminer l'une des trois déclarations afin d'éliminer l'incohérence. Parmi les pistes à explorer dans ce type de situation, nous envisageons l'étude et l'intégration d'un indice de confiance attaché aux différentes sources d'ou proviennent les constituants des ontologies étudiées.

Le système ONTOAST a été conçu pour la définition et la manipulation d'un ensemble limité de relations spatiales et temporelles, souvent définies pour des contextes "idéaux". C'est le cas, par exemple, des relations de distance euclidiennes définies pour des espaces isotropes. En réalité les relations de distance utilisées dans des applications sont bien plus complexes. Ainsi, on peut mesurer la distance entre deux points, par rapport au temps de voyage effectué sur différents réseaux de transports (trains, avion, voiture, *etc.*). De plus, la durée de voyage entre deux points A et B est souvent variable en fonction de la fluidité du trafic selon le moment de la journée. Il se peut

également que la distance ne soit pas symétrique. Selon que l'on aille de *A* vers *B* ou de *B* vers *A*, le temps de voyage peut être différent en fonction de la topographie du trajet. Ainsi, une extension et une adaptation des relations proposées par ONTOAST, pour mieux modéliser la réalité dans des applications, sont envisagées comme perspectives à moyen terme.

D'autres ontologies décrivant des relations spatiales et temporelles qualitatives existent à ce jour. Il nous semble intéressant d'étudier le coût de l'adaptation du système ONTOAST pour la prise en charge de concepts et relations spatiaux et temporels définis par d'autres ontologies que *GeoRSS-Simple*, *OWL Time* et *QualitativeSpatialRelations*. Ainsi, se pose la question de savoir si les opérateurs ajoutés au LMA sont suffisants pour appuyer la déduction de ces nouvelles relations. Sans doute la réponse est-elle non. Aussi, serait-il intéressant d'envisager la définition de nouveaux opérateurs par le biais de services Web.

8.2.3 Perspectives à long terme

Comme perspective à long terme, nous souhaitons analyser l'adaptation pour OWL 2 DL des associations sémantiques de type $\rho - isoAssociated$ et $\rho - jointAssociated$, et des techniques de découverte de celles-ci. Ces associations sont d'une complexité plus importante que les associations $\rho - pathAssociated$ car elles impliquent la découverte de paires d'associations de type $\rho - pathAssociated$.

Toujours comme perspective à long terme, nous entendons étudier l'*analyse sémantique* dans le cadre d'alignement d'ontologies. En réalité, il est souvent possible que différentes ontologies décrivent des domaines qui se recouvrent, ou des points de vues différentes sur les mêmes domaines. Les correspondances qui existent entre les classes, les propriétés et les individus définis par différentes ontologies ainsi que les indices de confiance qui leur correspondent peuvent être découvertes à l'aide d'outils d'alignement. Dans ce type de scénario, nous envisageons une étude sur les modifications qui doivent être mises en place au niveau de notre *framework* d'analyse sémantique pour qu'il prenne en compte les alignements calculés.

Ainsi, il nous semble pertinent d'étudier la possibilité d'étendre un contexte donné, par rapport à un ensemble d'appariements définis pour les classes et les propriétés qu'il inclut. Par exemple, si une classe C est incluse dans le contexte C_{Them} d'une requête sémantique, et si, dans le cas le plus simple, l'on connaît une ou plusieurs classes $D_1, D_2, \ldots, D_k$ définies dans d'autres ontologies calculées comme étant équivalentes à C, avec les indices de confiance respectifs $w_1, w_2, \ldots, w_k$, alors on peut obtenir k contextes thématiques dérivés de C_{Them}, dont chaque poids est calculé à partir du poids de C_{Them} diminué par le degré de confiance w_i. Ces contextes dérivés et les poids qui leurs sont attachés peuvent être utilisés pour déterminer la pertinence des $onto - path$ qui traversent plusieurs ontologies.

Bibliographie

[1] Opengis Simple Features Specification for SQL, 1999.

[2] Opengis reference model, 2002.

[3] Uml 2.0 ocl specification, object constraint language, 2003.

[4] Uniform resource identifier (uri) : Generic syntax, 2005.

[5] Owl 2 web ontology language document overview, 19 May 2009.

[6] Hoolet - homepage, last modified 2004. http ://owl.man.ac.uk/hoolet/.

[7] Protege 2000. La création d'ontologies web sémantique avec protégé-2000.

[8] P. Agarwal. Ontological considerations in giscience. In *International Journal of Geographical Information Science*, volume 19, 2005.

[9] R. Akerkar. *Introduction to artificial intelligence*. Prentice Hall of India, 2005.

[10] B. Aleman-Meza, C. Halaschek, I. Budak Arpinar, and A. Sheth. Context-aware semantic association ranking. In *In proceedings of the First International Workshop on Semantic Web and Databases*, Berlin, Germany, 2003.

[11] B. Aleman-Meza, C. Halaschek, I. Budak Arpinar, C. Ramakrishnan, and A. Sheth. Ranking complex relationships on the semantic web. *IEEE Internet Computing*, 9(3), 2005.

[12] B. Aleman-Meza, Chris Halaschek, I. budak Arpinar, and Amit Sheth. Context-aware semantic association ranking. Technical report, LSDIS Lab, University of Georgia, 2003.

[13] J. H. Alexander, M. J. Freiling, S. J. Shulman, J. L. Staley, S. Rehfus, and S. L. Messick. Knowledge level engineering : Ontological analysis. In *Proceedings of AAAI 86*, pages 963–968, Philadelphia, 1986.

[14] J.F. Allen. Maintaining knowledge about temporal intervals. In *Communication of the ACM*, pages 832–843, 1983.

[15] J. Alpert and N. Hajaj. We knew the web was big... The Official Google Blog, July 2008.

[16] T.A. Alspaugh. Thomas a. alspaugh's foundations material. http ://www.ics.uci.edu/ alspaugh/cls/shr/, September 2009. last modified September 2009.

[17] T. Amghar, D. Battistelli, and T. Charnois. Représenter le temps en langue dans le formalisme des graphes conceptuels une approche basée sur les schèmes sémantico-cognitifs. In *TALN 2001*, 2001.

[18] K. Anyanwu and A. Sheth. The rho operator : discovering and ranking associations on the semantic web. *ACM SIGMOD Record. SPECIAL ISSUE : Special section on semantic web and data management*, 31(4) :42 – 47, 2002.

[19] K. Anyanwu and A. Sheth. p-queries : Enabling querying for semantic associations on the semantic web. In *WWW2003*, Budapest, Hungary, 2003.

[20] K. Apt. *Principles of Constraint Programming*. Cambridge University Press, 2003.

[21] K. Apt and M. Wallace. *Constraint Logic Programming using ECLiPSe*. Cambridge University Press, 2006.

[22] K. R. Apt and S. Brand. A new constraint-based framework for qualitative reasoning. http ://homepages.cwi.nl/ apt/ps/ab05b.ps, 2005.

[23] M. A. Aufaure. Ontologies et fouille de données : quels bénéfices pour un web plus sémantique ? Seminaire ETIS, 2008.

[24] F. Azouaou, T. Cao, S. Dehors, C. Desmoulins, R. Dieng-Kuntz, and C. Farou-Zucker. Les outils du web semantique et du e-learning. In *Plate-forme Afia*, May 2005.

[25] A. Bailly. *Encivlopédie de géographie*, chapter Les representations en géographie, pages 369–383. Economica, 1995.

[26] J.A. Bateman, R.T. Kasper, J.D. Moore, and R.A. Whitney. A general organization of knowledge for natural language processing : the penman upper model. USC/Information Sciences Institute, Marina del Rey, CA, 1990.

[27] G. Baudelle and H. Regnault. Echelles et temporalités en géographie. *SEDES-DIEM*, 2004.

[28] S. Bechhofer, I. Horrocks, C. Goble, and R. Stevens. Oiled- a reasonable ontology editor for the semantic web. In *KI-2001. Advances in Artificial Intelligence*, pages 396– 408. Springer, 2001.

[29] S. Bechhofer, F. van Harmelen, J. Hendler, I. Horrocks, D. L. McGuinness, P. F. Patel-Schneider, and L. A. Stein. Owl web ontology language reference, w3c recommendation 10 february 2004, february 2004.

[30] B. Bennett and P. Agarwal. Semantic categories underlying the meaning of place. In *COSIT 2007*, pages 78 – 95, 2007.

[31] B. Berendt, A. Hotho, and G. Stumme. Towards semantic web mining. In Springer-Verlag, editor, *ISWC 2002*, pages 264–278, 2002.

[32] T. Berners-Lee. Semantic web road map. http ://www.w3.org/DesignIssues/Semantic.html, September 1998.

[33] T. Berners-Lee. Www past & future. http ://www.w3.org/2003/Talks/0922-rsoc-tbl/, 2003.

[34] T. Berners-Lee, J. A Hendler, and O. Lassila. The semantic web. *Scientific American*, Volume 1 :34–43, 2001.

[35] S. Berretti, A. Del Bimbo, and E. Vicario. Weighted walkthroughs between extended entities for retrieval by spatial arrangement. *IEEE Transactions on Multimedia*, 5(1) :52–70, 2003.

[36] J. Bertin. *La sémiologie graphique*. Paris, Gauthiers-Villars, 1967.

[37] C. Bessière, J. Euzenat, R. Jeansoulin, G. Ligozat, and S. Schwer. Raisonnement spatial et temporel. In *Actes 6e journées nationales PRC-GDR intelligence artificielle*, pages 77–88, Grenoble , France, 1997.

[38] P. V. Biron and A. Malhotra. XML Schema Part 2 : Datatypes Second Edition. W3C Recommendation 28 October 2004, 2004.

[39] I. Bloch. Fuzzy relative position between objects in image processing : A morphological approach. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(7) :657–664, 1999.

[40] D. Bourigault, N. Aussenac-Gilles, and J. Charlet. Construction de ressources terminologiques ou ontologiques à partir de textes : un cadre unificateur pour trois études de cas. *Revue d'Intelligence Artificielle,*, 18(4), 2004.

[41] S. Brand. Relation variables in qualitative spatial reasoning. In *KI-2004 : 27th German Conference on Artificial Intelligence*. Springer LNAI 3238, 2004.

[42] S. Brand. *Rule-based Constraint Propagation-Theory and Applications.* PhD thesis, University of Amsterdam, 2004.

[43] T. Braya, J. Paoli, C. M. Sperberg-McQueen, E. Maler, and F. Yergeau. Extensible markup language (xml) 1.0 (fifth edition). w3c recommendation 26 november 2008, 2008.

[44] D. Brickley and R.V. Guha. Resource description framework (RDF) schema specification 1.0, March 2000.

[45] C. Bruley, P. Genoud, and V. Dupierris. Guide utilisateur arom v2. Technical report, Rapport Interne INRIA, 2004.

[46] D. Calvanese, G. De Giacomo, M. Lenzerini, R. Rosati, and G. Vetere. Dl-lite : Practical reasoning for rich dls. 104, 2004.

[47] T. D. Cao. *Exploitation du web sémantique pour la veille technologique*. PhD thesis, Universite de Nice-Sophia Antipolis, 2006.

[48] C. Capponi. Un systeme de types pour la representation des connaissances par objets. *Revue d'Intelligence Artificielle*, 12(3) :309–344, 1998.

[49] C. Capponi, J. Gensel, and M. Page. Arom semantics. ftp ://ftp.inrialpes.fr/pub/romans/logiciels/arom/semantics.pdf, 2000.

[50] B. Carré. *Méthodologie orientée-objet pour la representation des connaissances. Concepts de points de vue, de representation multiple et évolutive d'objet*. PhD thesis, Université des Sciences et Techniques de Lille Flandres Artois, 1989.

[51] J. Carroll and J. Pan. Xml schema datatypes in rdf and owl. w3c working group note 14 march 2006, March 2006.

[52] R. Casati and A. Varzi. *Parts and Places : The Structures of Spatial Representations*. MIT Press, 1999.

[53] C. Cauvin. Cognitive and cartographic representations : Towards a comprehensive approach. In *Cybergeo, Cartographie, Imagerie, SIG*, 2002.

[54] CETIC. La création d'ontologies web sémantique avec protégé-2000.

[55] J. Charlet, B. Bachimont, and R. Troncy. Ontologies pour le web sémantique. *Information - Interaction - Intelligence. Une Revue en Sciences du Traitement de l'Information*, 2004.

[56] J. Charlet, B. Bachimont, and R. Troncy. *Le Web Semantique*. Cepadues, 2005.

[57] H. Chen, Y. Zou, L. Kagal, and T. Finin. F-owl : An owl inference engine in flora-2, 2003. http ://fowl.sourceforge.net/.

[58] Clark and Parsia. Pellet : The open source owl dl reasoner. Pellet home page, 2009.

[59] J. Clark. Xsl transformation (xslt) version 1.0, 1999.

[60] B. L Clarke. A calculus of individuals based on 'connection'. *Notre Dame Journal of Formal Logic*, 22(3) :204–218, July 1981.

[61] E. Clementini, P. Di Felice, and D. Hernández. Qualitative representation of positional information. *Artificial Intelligence*, 95 :317–356, 1997.

[62] E. Clementini and R. Laurini. Un cadre conceptuel pour modéliser les relations spatiales. *Revue des Nouvelles Technologies de l'Information (RNTI)*, vol. RNTI-E-14 :1–17, 2008.

[63] E. Clementini, J. Sharma, and M. Egenhofer. Modeling topological spatial relations : Strategies for query processing. *Computers and Graphics*, 18(6) :815–822, 1994.

[64] N. B. Cocchiarella. *Handbook of Metaphysics and Ontology*, chapter Formal Ontology, pages 640–647. Philosophia Verlag, 1991.

[65] A. G. Cohn. Qualitative spatial representation and reasoning techniques. In *KI*, pages 1–30, 1997.

[66] A.G. Cohn, B. Bennett, J.Gooday, and N. M. Gotts. *Spatial and temporal reasoning*, chapter Representing and reasoning with qualitative spatial relations about regions, pages 97–134. Kluwer, 1997.

[67] A.G. Cohn and S.M. Hazarika. Qualitative spatial representation and reasoning : An overview. *Fundamenta Informaticae*, 46 :1–29, January 2001.

[68] J. F. Condotta and E. Wurbel. *Raisonnement sur l'espace et le temps*, chapter Réseaux de contraintes temporelles et spatiales, pages 181–225. Lavoisier, 2007.

[69] O. Corby, R. Dieng, C. Faron-Zucker, F. Gandon, and A. Giboin. Le moteur de recherche sémantique corese. In *Proceedings of the Workshop Raisonner le web s´emantique avec des graphes, AFIA platform,*, 2005.

[70] M. Cristani. The complexity of reasoning about spatial congruence. *Journal of Artificial Intelligence Research*, 11 :361–390, 1999.

[71] O. Dameron. *Modelisation, representation et partage de connaissances anatomiques sur le cortex cerebral*. PhD thesis, Universite de Rennes I, 17 decembre 2003.

[72] M. de Kunder. The size of the world wide web, 2008.

[73] M. Dean, G. Schreiber, S. Bechhofer, F. van Harmelen, J. Hendler, I. Horrocks, D. McGuinness, P. Patel-Schneider, and L. Stein. Owl web ontology language reference, june 2005.

[74] R. Dechter. *Constraint Processing*. Morgan Kaufmann, 2003.

[75] L. Dekker. *FROME : représentation multiple et classification d'objets avec points de vue.* PhD thesis, Université des Sciences et Techniques de Lille Flandres Artois, 1994.

[76] J. Denegre and F. Salge. *Les Systemes d'Information Géographique*. Presses Universitaires de France, Paris 75006, 1996.

[77] L. Ding, T. Finin, A. Joshi, R. Pan, R. S. Cost, Y. Peng, P. Reddivari, V. Doshi, and J. Sachs. Swoogle : a search and metadata engine for the semantic web. In *Proceedings of the thirteenth ACM international conference on Information and knowledge management*, 2004.

[78] L. Ding, R. Pan, T. Finin, A. Joshi, Y. Peng, and P. Kolari. Finding and ranking knowledge on the semantic web. In *Proceedings of the 4th International Semantic Web Conference*, Galway IE, 2005. Springer-Verlag.

[79] J. F. Djoufak Kengue, J. Euzenat, and P. Valtchev. Alignement d'ontologies dirigé par la structure. 2008.

[80] S. Donikian. Raisonnement spatial.

[81] C. du Mouza and P. Rigaux. Bases de données spatio-temporelles. In *Documents et évolution, Cépaduès, publié dans le cadre de l'école thématique du GDR I3*, 2000.

[82] R. Ducournau. *Langages et modèles à objets : Etat des recherches et perspectives*, chapter La logique des objets : application à la classification incertaine., pages 351–380. INRIA, 1998.

[83] R. Ducournau. Des langages d'objets aux logiques de description : les systèmes classificatoires. Technical report, LIRMM, 2000.

[84] R. Ducournau, J. Euzenat, G. Masini, and A. Napoli. *Langages et modèles à objets. Etat des recherches et perspectives.* INRIA, 1998.

[85] M. Egenhofer and R. Franzosa. Point-set topological spatial relations. *International Journal of Geographical Information Systems*, 5(2) :161–174, 1991.

[86] M. Egenhofer, M.A. Rodriguez, and A. Blaser. Query pre-processing of topological constraints : Comparing a composition-based with neighborhood-based approach. *SSTD*, pages 362–379, 2003.

[87] M. J. Egenhofer. Reasoning about binary topological relations. In *Proceedings of the Second Symposium on Large Spatial Databases SSD'91*, volume 525 of *Lecture Notes in Computer Science*, pages 143–160. Springer-Verlag, Berlin, Heidelberg, New York, 1991.

[88] M. J. Egenhofer. Towards the semantic geospatial web. *ACM GIS 2002*, 1 :34–43, 2002.

[89] M.J. Egenhofer and D. M. Mark. Spatial information theory : A theoretical basis for gis - international conference. In A. U. Frank and W. Kuhn, editors, *COSIT'95*, pages 1–15. Springer-Verlag, 1995.

[90] M.J. Egenhofer and J. Sharma. Topological relations between regions in r2 and z2. In Springer-Verlag, editor, *Advances in spatial databases- third international symposium on large spatial databases*, volume 692, pages 316–336, Singapore, 1993.

[91] J. Euzenat and P. Shvaiko. *Ontology matching*. Springer-Verlag, 2007.

[92] R. Fikes, P. Hayes, and I. Horrocks. Owl-ql - a language for deductive query answering on the semantic web. *Web Semantics : Science, Services and Agents on the World Wide Web*, 2(1) :19–29, 2004.

[93] F. Fonseca and A. Sheth. The geospatial semantic web. research priority white paper. Technical report, University Consortium for Geographic Information Science, 2003.

[94] K. D. Forbus. Qualitative process theory. *Artificial Intelligence*, 24 :85–168, 1984.

[95] Python Software Foundation. Python patterns - implementing graphs. http ://www.python.org/doc/essays/graphs/, 2003.

[96] A. U. Frank. Qualitative spatial reasoning : Cardinal directions as an example. *International Journal of Geographical Information Science*, 10 :269–290, 1996.

[97] C. Freksa. *Theories and Methods of Spatio-Temporal Reasoning in Geographic Space*, chapter Using orientation information for qualitative spatial reasoning. Lecture Notes in Computer Science. Springer-Verlag, Berlin, Heidelberg, New York, 1992.

[98] Y. Gargouri, B. Lefebvre, and J.G. Meunier. Maintenance des ontologies à partir d'analyses textuelles. In *71e congrès de l'association canadienne-française pour l'avancement des sciences*, 2003.

[99] P. Gatalsky, N. Adrienko, and G. Adrienko. Interactive analysis of event data using space-time cube. *Information Visualisation*, IEEE :145–152, 2004.

[100] J. Gensel. *Contraintes et représentation de connaissances par objets - Application au modèle Tropes*. PhD thesis, Université Joseph Fourier, 1995.

[101] J. Gensel. Habilitation à diriger des recherches : Représentation de connaissances par objets pour la modélisation et l'adaptation d'informations multimédias et spatio-temporelles. 2006.

[102] J. Gensel, C. Capponi, P. Genoud, and D. Ziébelin. Vers une intégration des relations partie-tout en arom. In *Langages et Modèles à Objets (LMO 2006)*, 2006.

[103] A. Gómez-Pérez and D. Manzano-Macho. Ontoweb d.1.5 a survey of ontology learning methods and techniques. Technical report, Universidad Politécnica de Madrid, 2003.

[104] C. Golbreich and E. K. Wallace. Owl 2 web ontology language : New features and rationale., 2009.

[105] P. Gomez. A survey of ontology tools, deliverable 1.3, ontoweb, May 2002.

[106] N.M. Gotts. Formalising commonsense topology : The inch calculus. In *Proc. Fourth International Symposium on Artificial Intelligence and Mathematics*, 1996.

[107] R. Goyal. *Similarity assessment for cardinal directions between extended spatial objects*. PhD thesis, The University of Maine, May, 2000.

[108] R. Goyal and M.J. Egenhofer. The direction-relation matrix : A representation for directions relations between extended spatial objects. In *The Annual Assembly and the Summer Retreat of University Consortium for Geographic Information Systems Science*, 1997.

[109] R. Goyal and M.J. Egenhofer. Cardinal directions between extended spatial objects. *IEEE Trans. Data Knowledge Engrg.*, 2000.

[110] B. C. Grau, I. Horrocks, B. Motik, B. Parsia, P. Patel-Schneider, and U. Sattler. Owl2 : The next step forowl. Elsevier, 2008.

[111] M. Grigni, D. Papadias, and C. Papadimitriou. Topological inference. In *IJCAI-95*, pages 901–906, 1995.

[112] T. Gärling and R.G. Golledge. Environmental perception and cognition. *Advances in environment, behavior, and design*, 2 :203–236, 1987.

[113] T. Gruber. Toward principles for the design of ontologies used for knowledge sharing. *International Journal Human-Computer Studies*, 43 :907–928, november 1995.

[114] T. R. Gruber. *Formal Ontology in Conceptual Analysis and Knowledge Representation*, chapter Toward Principles for the Design of Ontologies used for Knowledge Sharing. Kluwer Academic Publishers, 1993.

[115] S. Grumbach, P. Rigaux, and L. Segoufin. Spatio-temporal data handling with constraints. In *In Intl. Symposium on Geographic Information Systems (ACM-GIS'98)*. Morgan Kaufmann, 1998.

[116] A. Grzegorczyk. Undecidability of some topological theories. *Fundamenta Mathematicae*, 38 :137–152, 1951.

[117] N. Guarino. Understanding, building, and using ontologies, a commentary to using explicit ontologies in kbs development, 1997.

[118] N. Guarino and P. Giaretta. *Ontologies and Knowledge Bases Towards a Terminological Clarification*, pages 25–32. IOS Press, Amsterdam, IOS press edition, 1995.

[119] N. Guarino and R. Poli. Formal ontology in conceptual analysis and knowledge representation. *Special issue of the International Journal of Human and Computer Studies*, 43(5/6) :625–640, 1995.

[120] C. Gutierrez, C. Hurtado, and A. Vaisman. Temporal rdf. In *Second European Semantic Web Conference*, Heraklion, Crete, Greece, 2005.

[121] R.H. Guting. An introduction to spatial database systems. *The VLDB Journal*, 3(4) :357–400, 1994.

[122] V. Haarslev, C. Lutz, and R. Moller. Foundations of spatioterminological reasoning with description logics. In *Principles of Knowledge Representation and Reasoning : Proceedings of the Sixth International Conference (KR98)*, pages 112–123, 1998.

[123] F. Hakimpour, B. Aleman-Meza, M. Perry, and A. Sheth. Data processing in space, time and semantics dimensions. In *Terra Cognita 2006 - Directions to the Geospatial Semantic Web*, 2006.

[124] C. Halaschek, B. Aleman-Meza, B.I. Arpinar, and A.P. Sheth. Discovering and ranking semantic associations over a large rdf metabase. In *Proceedings of the 30th VLBD Conference*, Toronto, Canada, 2004.

[125] S. Handschuh, S. Staab, and F. Ciravegna. S-cream - semi-automatic creation of metadata. In *Proceedings of the 13th International Conference on Knowledge Engineering and Management*. Springer, 2002.

[126] S. Hawke. Surnia - homepage, 2003. http ://www.w3.org/2003/08/surnia/.

[127] P. Hayes and B. McBride. Rdf semantics. w3c recommendation 10 february 2004, 2004.

[128] D. Hernandez, E. Clementini, and P. Di Felice. Qualitative distances, 1995.

[129] J. R. Herring. The opengis abstract specification topic 1 : Feature geometry (iso 19107 spatial schema), 2001.

[130] J.R. Hobbs, W. Croft, T. Davies, E. Douglas, and K. Laws. Commonsense metaphysics and lexical semantics. *Computational Linguistics*, 13 :241–250, 1987.

[131] J.R. Hobbs and J.Z. Pan. An ontology of time for the semantic web. In *ACM Transactions on Asian Language Information Processing*, volume 3, pages 66–85, 2005.

[132] J.R. Hobbs and J.Z. Pan. Time ontology in owl. w3c working draft 27 september (2006). W3C Working Draft 27 September (2006), 2006.

[133] I. Horrocks. The fact system - homepage. http ://www.cs.manchester.ac.uk/ horrocks/FaCT/, 2003. http ://www.cs.manchester.ac.uk/ horrocks/FaCT/.

[134] I. Horrocks, P. F. Patel-Schneider, H. Boley, S. Tabet, B. Grosof, and M. Dean. Swrl : A semantic web rule language combining owl and ruleml. w3c member submission 21 may 2004, 2004.

[135] I. Horrocks, P. F. Patel-Schneider, and F. van Harmelen. From SHIQ and RDF to OWL : the making of a web ontology language. *Journal of Web Semantics*, 1 :7–26, 2003.

[136] NIMA (National Imagery and Mapping Agency). Department of defense world geodetic system 1984. its definition and relationships with local geodetic systems. Technical report, NIMA (National Imagery and Mapping Agency), January 2000.

[137] T. Ishikawa and D. R. Montello. Spatial knowledge acquisition from direct experience in the environment : Individual differences in the development of metric knowledge and the integration of separately learned places. *Cognitive Psychology*, 52 :93–129, 2006.

[138] A. Kiryakov, B. Popov, D. Ognyanoff, D. Manov, A. Kirilov, and M. Goranov. Semantic annotation, indexing, and retrieval. In *Proceedings of the 2nd International Semantic Web Conference*, pages 484–499, 2003.

[139] G. Klose, E. Lang, and T. Pirlein. *Ontologie und Axiomatik der Wissensbasis von LILOG*. Springer-Verlag, Berlin, 1992.

[140] G. Klyne and J. J. Carroll. Resource description framework (RDF) concepts and abstract syntax, February 2004.

[141] R. Kosala and H. Blockeel. Web mining research : A survey. In *SIGKDD Explorations*, june 2000.

[142] A. Krokhin, P. Jeavons, and P. Jonsson. Reasoning about temporal relations : The tractable subalgebras of allen's interval algebra. *Journal of the ACM*, 50(5) :591–640, 2003.

[143] X. Lacot. Introduction à owl, un langage xml d'ontologies web, Juin 2005.

[144] O. Lassila and R.R. Swick. Resource description framework (RDF) model and syntax specification, 1999.

[145] P. Laublet, C. Reynaud, and J. Charlet. Sur quelques aspects du web semantique. 2003.

[146] A.K. Lautenschutz, C. Davies, M. Raubal, A. Schwering, and E. Pederson. The influence of scale, context and spatial preposition in linguistic topology. In *5th International Conference Spatial Cognition 2006*, pages 439–452, 2007.

[147] F. Le Ber, L. Mangelinck, and A. Napoli. Étude de treillis de relations topologiques pour l'interprétation d'images satellitaires. *Revue internationale de Géomatique*, 11 :215–249, 2001.

[148] F. Le Ber and A. Napoli. The design of an object-based system for representing and classifying spatial structures and relations. *Journal of Universal Computer Science*, 8(8) :751–773, 2002.

[149] F. le Ber and A. Napoli. *Raisonnements sur l'espace et le temps*, chapter Treillis pour le raisonnement spatial, pages 225–249. Lavoisier, 2007.

[150] J. Leskovec and E. Horvitz. Planetary-scale views on a large instant-messaging network. In *WWW 2008*, Beijing, China, 2008.

[151] J. Lieberman. Geospatial semantic web interoperability experiment report. Technical report, Open Geospatial Consortium Inc., 2006.

[152] J. Lieberman, R. Singh, and C. Goad. W3c geospatial vocabulary. w3c incubator group report 23 october 2007.

[153] J. Lieberman, R. Singh, and C. Goad. W3c geospatial ontologies. w3c incubator group report 23 october 2007, 2007.

[154] G. Ligozat. Tractable relations in temporal reasoning : pre-convex relations. In *Proceedings of the ECAI'94 Workshop on Spatial and Temporal Reasonig*, Amsterdamn, Pays-Bas, 1994.

[155] G. Ligozat. Reasoning about cardinal directions. *Journal of Visual Languages and Computing*, 9 :23–44, 1998.

[156] M. Liu, G. Dobbie, and T. W. Ling. A logical foundation for deductive object-oriented databases. *ACM Transactions on Database Systems*, 27(1) :117 – 151, 2002.

[157] P. Longley, M. Goodchild, D. Maguire, and D. Rhind. *Geographic Information Systems and Science*. Wiley, 2005.

[158] C. Lutz. *The Complexity of Reasoning with Concrete Domains*. PhD thesis, LuFG Theorethical Computer Science, RWTH Aschen, Germany, 2002.

[159] A. K. Mackworth. Consistency in networks of relations. *Artif. Intell.*, 8(1) :99–118, 1977.

[160] A. Maedche and S. Staab. Ontology learning for the semantic web. In *IEEE Intelligent Systems. Special Issue on the Semantic Web*, volume 16(2), 2001.

[161] Alexander Maedche and Steffen Staab. *Handbook on ontologies*, chapter Ontology learning. Springer, 2004.

[162] F. Manola and E. Miller. RDF primer, February 2004.

[163] B. McBride. Jena : Implementing the rdf model and syntax specification. In *Semantic Web Workshop WWW2001*, 2001.

[164] J.E. McGrath and F. Tschan. *Temporal Matters in Social Psychology : Examining the Role of Time in the Lives of Groups and Individuals*. Psychological Association, Waschington DC, 2004.

[165] D. L. McGuinness and F. van Harmelen. OWL web ontology language overview, February 2004.

[166] mindswap. Pallet, 2003.

[167] A. D. Miron, C. Capponi, J. Gensel, M. Villanova-Oliver, D. Ziebelin, and P. Genoud. Rapprocher arom de owl... In *LMO*, pages 99–116, 2007.

[168] A.D. Miron. Représentation de connaissances par objets pour le web sémantique ou comment rapprocher arom de owl. Master's thesis, Université Joseph Fourier, 2006.

[169] A.D. Miron, J. Gensel, and M. Villanova-Oliver. Semantic analysis for the geospatial web - application to owl-dl ontologies. In *8h International Symposium on Web and Wireless GIS (W2GIS 2008)*, Shanghai, China, Decembre 2008.

[170] A.D. Miron, J. Gensel, and M. Villanova-Oliver. Analytique semantique spatio-temporelle pour les ontologies owl-dl. In *EGC*, pages 379–390, 2009.

[171] A.D. Miron, J. Gensel, and M. Villanova-Oliver. Discovery of spatial links between individuals on the geospatial semantic web. In *Geoinformatics*, 2009.

[172] A.D. Miron, J. Gensel, M. Villanova-Oliver, and H. Martin. Towards the geo-spatial querying of the semantic web with ontoast. Cardiff, UK, 2007.

[173] A.D. Miron, J. Gensel, M. Villanova-Oliver, and H. Martin. Vers le web semantique geospatial avec ontoast. In *Atelier "Representation et raisonnement sur le temps et l'espace", plate-forme AFIA 2007*, Grenoble, France, 2007.

[174] B. Moisuc. *Conception et Mise en OEuvre de Systèmes d'Information Spatio-Temporelle Adaptatifs : le framework ASTIS*. PhD thesis, Université Joseph Fourier, 2007.

[175] B. Moisuc, C. Capponi, P. Genoud, J. Gensel, and D. Ziébelin. Modélisation algébrique et représentation de connaissances par objets en arom. In *LMO 2007*, pages 83–98, 2007.

[176] B. Moisuc, P. Genoud, D. Ziébelin, J. Gensel, and C. Capponi. Apports de la modélisation algebrique pour la représentation de connaissances par objets : illustration en arom. In *17e journees francophones d'Ingenierie des connaissances*, 2006.

[177] D. R. Montello. Scale and multiple psychologies of space. In *Spatial information theory : A theoretical basis for GIS*, Lecture Notes in Computer Science, pages 312–321, Berlin, 1993. Springer - Verlag.

[178] B. Motik. On the properties of metamodeling in owl. In Springer-Verlag Berlin Heidelberg, editor, *LNCS 3729*, volume ISWC2005, pages 548–562, 2005.

[179] B. Motik and I. Horrocks. Owl datatypes : Design and implementation. In Amit P. Sheth, Steffen Staab, Mike Dean, Massimo Paolucci, Diana Maynard, Timothy Finin, and Krishnaprasad Thirunarayan, editors, *Proc. of the 7th Int. Semantic Web Conference (ISWC 2008)*, volume 5318 of *LNCS*, pages 307–322, Karlsruhe, Germany, 2008. Springer.

[180] B. Motik, P. F. Patel-Schneider, B. C. Grau, I. Horrocks, B. Parsia, and U. Sattler. Owl 2 web ontology language direct semantics, 2009.

[181] B. Motik, P. F. Patel-Schneider, B. Parsia, C. Bock, A. Fokoue, P. Haase, R. Hoekstra, I. Horrocks, A. Ruttenberg, U. Sattler, and M. Smith. Owl 2 web ontology language structural specification and functional-style syntax, 2009.

[182] B. Motik, Peter F. Patel-Schneider, and I. Horrocks. Owl 1.1 web ontology language structural specification and functional-style syntax, May 2007.

[183] P. Muller and V. Dugat. *Raisonnement sur l'espace et le temps*, chapter Répresentation en logique classique. Lavoisier, 2007.

[184] N. B. Mustapha, M. A. Aufaure, H. B. Zghal, and H. B. Ghezala. Recherche semantique basee sur les ontologies modulaires et le raisonnement à base de cas. In *Actes des 20es Journees Francophones d'Ingénierie des Connaissances*, 2009.

[185] A. Napoli. Ontologies, fouille de données et web sémantique, 2003.

[186] A. Napoli, B. Carre, R. Ducournau, J. Euzenat, and F. Rechenmann. Objets et représentation, un couple en devenir. *RSTI-L'objet*, pages 61–81, 2004.

[187] B. Nebel and H.J. Burckert. Reasoning about temporal relations : A maximal tractable subclass of allen's interval algebra. *Journal of the Association of Computing Machinery*, 42 :43–66, 1995.

[188] BBC News. The tech lab : Bradley horowitz. bradley horowitz, responsible for novel technology development at search giant yahoo, looks ahead to the "internet of things"., 2007.

[189] N. Noy, A. Rector, P. Hayers, and C. Welty. Defining n-ary relations on the semantic web, April 2006.

[190] D. O'Dea, S. Geoghegan, and C. Ekins. Dealing with geospatial information in the semantic web. In *In Proc. Australasian Ontology Workshop (AOW 2005)*, pages 69–73, Sydney, Australia, 2005.

[191] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking : Bringing order to the web. In *Proceedings of the 7th International World Wide Web Conference*, pages 161–172, Brisbane, Australia, 1998.

[192] M. Page, J. Gensel, C. Capponi, C. Bruley, P. Genoud, and D. Ziébelin. Représentation de connaissances au moyen de classes et d'associations : le système arom. In *Actes de LMO'00*, pages 91–106, Mont Saint-Hilaire, Canada, 2000.

[193] M. Page, J. Gensel, C. Capponi, C. Bruley, P. Genoud, D. Ziebelin, D. Bardou, and V. Dupierris. A new approach in object-based knowledge representation : the arom system. *Lecture Notes in Artificial Intelligence*, pages 113–118, 2001.

[194] F. Pan. *Representing Complex Temporal Phenomena for the Semantic Web and Natural Language*. PhD thesis, University of Southern California, December 2007.

[195] J. Z. Pan. *Description Logics : reasoning support for the Semantic Web*. PhD thesis, University of Manchester, 2004.

[196] C. Patel, K. Supekar, Y. Lee, and E. K. Park. Ontokhoj : a semantic web portal for ontology searching, ranking and classification. In *Proceedings of the 5th ACM international workshop on Web information and data management*, pages 58–61, 2003.

[197] P. F. Patel-Schneider, P. Hayes, and I. Horrocks. Owl web ontology language semantics and abstract syntax. w3c recommendation 10 february 2004, february 2004.

[198] P. F. Patel-Schneider and I. Horrocks. Owl 1.1 web ontology language overview, 23 May 2007.

[199] M. Perry and A. Sheth. A framework to support spatial, temporal and thematic analytics over semantic web data. Technical Report Technical Report KNOESIS-TR-2008-01, 2008.

[200] M. Perry, A. Sheth, F. Hakimpour, and P. Jain. What, where and when : Supporting semantic, spatial and temporal queries in dbms. Technical report, KNOESIS-TR-2007-01, 2007.

[201] D. Peuquet. *Spatial and temporal reasoning*, chapter Representations of space and time. New York : Guilford, 2002.

[202] I. Pratt and D. Schoop. A complete axiom systme for polygonal mereotopology or the real plane. *Journal of Philosophical Logic*, pages 621–658, 1998.

[203] Pew Internet & American Life Project. Online activities - total. the percentage of internet users who have ever done an online activity., Janvier 2009.

[204] E. Prud'hommeaux and A. Seaborne. Sparql query language for rdf, January 2008.

[205] Racer home page. What is racerpro ? an overview, 2009.

[206] E. Rahm and P. Bernstein. A survey of approaches to automatic schema matching. *VLDB Journal*, 10 :334–350, 2001.

[207] D. A. Randell, Z. Cui, and A. G. Cohn. A spatial logic based on regions and connection. In *3rd International Conference on Knowledge Representation and Reasoning*. Morgan Kaufmann, 1992.

[208] F. Rechenmann, P. Fontanille, and P. Uvietta. Shirka : Système de gestion de bases de connaissances centrées-objets. INRIA et Laboratoire Artémis IMAG, 1990. Grenoble, France.

[209] J. Renz. Qualitative spatial reasoning with topological information. *LNAI*, 2293 :31–40, 2002.

[210] J. Renz and B. Nebel. *Spatial Cognition*, volume 1404/1998, chapter Spatial Reasoning with Topological Information, pages 351–371. Lecture Notes in Computer Science, 1998.

[211] G. Retz-Schmidt. Various views on spatial prepositions. *AI Magazine*, 9(2) :95–105, 1998.

[212] Anne Ruas. *Le changement de niveau de détail dans la représentation de l'information géographique*. Habilitation à diriger des recherches, Université de Marne-La-Vallée, 2004.

[213] D. Schell. Geoscinetist's geospatial web requiers tochnology convergence. *Géosciences*, 6, 2007.

[214] M. Schneider, J. Carroll, I. Herman, and P. F. Patel-Schneider. Owl 2 web ontology language rdf-based semantics, 2009.

[215] R. Sedgewick. Finding paths in graphs. http ://www.cs.princeton.edu/ rs/talks/PathsInGraphs07.pdf, 2007. Princeton University.

[216] A. Serres. Recherche d'information sur internet : ou en sommes-nous, ou allons-nous ?, 2004.

[217] J. Sharma, D. M. Flewelling, and M. J. Egenhofer. A qualitative spatial reasoner. In *In the Proceedings of the 6th International Symposium on Spatial Data Handling (SDH*, pages 665–681, 1994.

[218] A. Sheth, B. Aleman-Meza, I. B. Arpinar, C. Bertram, Y. Warke, C. Ramakrishnan, C. Halaschek, K. Anyanwu, D. Avant, F. S. Arpinar, and K. Kochut. Semantic association identification and knowledge discovery for national security applications. *Journal of Database Management on Database Technology*, 16(1) :33–53, 2005.

[219] A. Sheth, B.I. Arpinar, and V. Kashyap. *Enhancing the Power of The Internet : Studies in Fuzziness and Soft Computing*, chapter Relationships at the Heart of Semantic Web : Modeling, Discovering, and Exploiting Complex Semantic Relations. 2002.

[220] P. Simons. *Parts : A Study in Ontology*. Clarendon Press, Oxford, 1987.

[221] E. Sirin, B. Parsia, B. C. Grau, A. Kalyanpur, and Y. Katz. Pellet : A practical owl-dl reasoner. *Journal of Web Semantics*, 5(2), 2007.

[222] S. Skiadopoulos, C. Giannoukos, N. Sarkas, P. Vassiliadis, T. Sellis, and M. Koubarakis. Computing and managing cardinal direction relations. *IEEE Transaction on Knowledge and Data Engineering*, 17(12) :1610–1623, 2005.

[223] S. Skiadopoulos and M. Koubarakis. Composing cardinal direction relations. *Artificial Intelligence*, 152 :143–171, 2004.

[224] S. Skiadopoulos, N. Sarkas, T. Sellis, and M. Koubarakis. A family of directional relation models for extended objects. *IEEE Transactions on knowledge and data engineering*, 19 :1116–1130, 2007.

[225] M. Smith, C. Welty C., and D. McGuinness. Owl web ontology language guide. w3c recommendation, February 2004.

[226] S. Spaccapietra, N. Cullot, C. Parent, and C. Vangenot. On spatial ontologies, 2004.

[227] l'utilité linguistique Spirit.

[228] P. Spyns, R. Meersman, and M. Jarrar. Data modelling versus ontology engineering. *ACM SIGMOD Record : Special section on semantic web and data management*, 31 :12–17, 2002.

[229] E. Thomas, J. Z. Pan, and D. Sleeman. Ontosearch2 : Searching ontologies semantically. In *Proc. of OWL : Experiences and Directions (OWLED 2007)*, 2007.

[230] D. Tsarkov and I. Horrocks. Fact++ description logic reasoner : System description. In *Int. Joint Conf. on Automated Reasoning*, volume 4130 of *Lecture Notes in Artificial Intelligence*, pages 292–297. Springer, 2006.

[231] V. Uren, P. Cimiano, J. Iria, S. Handschuh, M. Vargas-Vera, E. Motta, and F. Ciravegna. Semantic annotation for knowledge management : Requirements and a survey of the state of the art. *Journal of Web Semantics*, 2005.

[232] W. Viana, S. Hammiche, B. Moisuc, M. Villanova-Oliver, J. Gensel, and H. Martin. Semantic keyword-based retrieval of photos taken with mobile devices. In *MoMM*, pages 192–199, 2008.

[233] W. Viana, A. Miron, B. Moisuc, J. Gensel, M. Villanova-Oliver, and H. Martin. Towards semantic and context-aware management of mobile multimedia. *special issue of Multimedia Tools and Applications*, 2009.

[234] W. Viana, M. Villanova-Oliver, J. Gensel, and H. Martin. Annotation contextuelle automatique avec photomap. In *UbiMob'08*, Saint-Malo, France, Mai 2008.

[235] M. Vilain and H. Kautz. Constraint propagation algorithms for temporal reasoning. In *Proceedings of the 5th National Conference on Artificial Intelligence (AAAI'86)*, San Francisco, 1986. Morgan Kaufmann.

[236] L. Visser. Iso 8601 : 2000(e) data elements and interchange formats - information interchange - representation of dates and times. second edition 2000-12-15, 2000.

[237] C. A. Welty and N. Guarino. Supporting ontological analysis of taxonomic relationships. *Data Knowledge Engineering*, 39(1) :51–74, 2001.

[238] K. Wilkinson, C. Sayers, H. Kuno, and D. Reynolds. Efficient rdf storage and retrieval in jena2. Technical report, Enterprise Systems and Data Management Laboratory HP Laboratories Palo Alto, 2003.

[239] M. Wolf and C. Wicksteed. Date and time formats., 1997.

[240] S. Zghal, S. B. Yahia, E. M. Nguifo, and Y. Slimani. Soda : Une approche structurelle pour l'alignement d'ontologies owl- dl. In *ZFO2007*, 2007.

ANNEXES

Annexe A

La syntaxe du langage AROM-ONTO

A.1 BNF du langage AROM-ONTO

```
kb ::=
     KNOWLEDGE_BASE IDF
     import_statements
     documentations
     kb_predefined_variables
     kb_components;

import_statements ::=
     IMPORT imported_kb;

imported_kb ::=
     \\vide | IDF | imported_kb COMMA IDF;

kb_predefined_variables ::=
     VERSION_INFO TYPE STRING
     SEE_ALSO TYPE STRING
     IS_DEFINED_BY TYPE STRING
     LABEL TYPE STRING
     PRIOR_VERSION TYPE IRI
     INCOMPATIBLE_WITH TYPE IRI;

documentations ::=
     // vide | documentations documentation ;

documentation ::=
     DOCUMENTATION (STRING_LITERAL | LONG_LITERAL | URL_LITERAL);

kb_components ::=
     // vide | kb_components kb_component ;

kb_component ::=
     class | association | property | instance | tuple | axioms;

class ::=
     CLASS IDF
```

```
     super_class_descriptor
     documentations
     class_variables_and_composition ;



super_class_descriptor ::=
     // vide | SUPER_CLASS super_class_list ;

super_class_list ::=
     class_expression | class_expression COMMA super_class_list ;

class_variables_and_composition ::=
     // vide | VARIABLES variables ;

variables ::=
     // vide | variables variable ;

variable ::=
     VARIABLE IDF
     variable_descriptors ;

variable_descriptors ::=
     // vide | variable_descriptors variable_descriptor ;

variable_descriptor ::=
     TYPE type
     | DOMAIN domain
     | CARDINALITY variable_cardinality
     | DEFAULT value
     | documentations
     | UNIT STRING_LITERAL
     | ATTACHMENT (BOOLEAN_LITERAL | attachment)
     | DEFINITION equations ;

attachment ::=
     IDF classnames COLON IDF ;

classnames ::=
     // vide | classnames classname ;

classname ::=
     DOT IDF;

variable_cardinality ::=
```

```
     min_bound max_bound ;

min_bound ::=
     // vide | MIN INTEGER_LITERAL ;

max_bound ::=
     // vide | MAX INTEGER_LITERAL | MAX MULT ;

type ::=
     base_type | new_type | SET_OF base_type | LIST_OF base_type
     | SET_OF new_type| LIST_OF new_type ;

base_type ::=
     INTEGER | FLOAT | BOOLEAN | STRING | IDF ;

new_type ::=
 IRI
 | POINT | LINE | LINEARRING | POLYGON | MULTILINE | MULTIAREA
 | INSTANT | INTERVAL | DURATION | MULTIINTERVAL | MULTIINSTANT;

domain ::=
     domain_set | domain_interval ;

domain_interval ::=
     LBRACK (value | MULT) DOTDOT (value | MULT) RBRACK ;

domain_set ::=
     LCURL values RCURL ;

values ::=
     value | values COMMA value ;

value ::=
     literal
     | LCURL RCURL
     | LCURL literals RCURL
     | LBRACK RBRACK
     | LBRACK literals RBRACK ;

literals ::=
     literal
     | literals COMMA literal;

literal ::=
     INTEGER_LITERAL
```

```
     | MINUS INTEGER_LITERAL
     | FLOAT_LITERAL
     | MINUS FLOAT_LITERAL
     | string_literal
     | BOOLEAN_LITERAL
     | IDF ;

class_expression ::=
     IDF
     |intersection_class
     |union_class
     |complement_class
     |enumerated_class
     |some_restriction
     |all_restriction
     |has_value_restriction
     |has_self_restriction
     |min_cardinality
     |max_cardinality
     |exact_cardinality
     |some_values_in
     |all_values_in
     |value_equals
     |variable_min_cardinality
     |variable_max_cardinality
     |variable_exact_cardinality
     super_class_descriptor
     documentations;

some_values_in ::=
     SOME_VALUES LPAREN variable_list domain RPAREN ;

all_values_in ::=
     ALL_VALUES LPAREN variable_list domain RPAREN;

variable_list ::=
     IDF |variable_list IDF;

value_equals ::=
     VALUE_EQUALS LPAREN IDF value RPAREN ;

variable_min_cardinality ::=
     VAR_MIN_CARD LPAREN IDF n RPAREN
     |VAR_MIN_CARD LPAREN IDF n domain RPAREN ;
```

```
variable_max_cardinality ::=
     VAR_MAX_CARD LPAREN IDF n RPAREN
     |VAR_MAX_CARD LPAREN IDF n domain RPAREN ;

variable_exact_cardinality ::=
     VAR_EXACT_CARD LPAREN IDF n RPAREN
     |VAR_EXACT_CARD LPAREN IDF n domain RPAREN ;

min_cardinality ::=
      MIN_CARDINALITY LPAREN INTEGER_LITERAL property_expression RPAREN
     |MIN_CARDINALITY LPAREN INTEGER_LITERAL
     property_expression class_expression RPAREN;

max_cardinality ::=
     MAX_CARDINALITY LPAREN INTEGER_LITERAL property_expression RPAREN
     |MAX_CARDINALITY LPAREN INTEGER_LITERAL
     property_expression class_expression RPAREN;

exact_cardinality ::=
     EXACT_CARDINALITY LPAREN INTEGER_LITERAL property_expression RPAREN
     |EXACT_CARDINALITY LPAREN INTEGER_LITERAL
     property_expression class_expression RPAREN ;

intersection_class ::=
     INTERSECTION_OF LPAREN class_list RPAREN;

union_class ::=
     UNION_OF LPAREN class_list RPAREN;

complement_class ::=
     COMPLEMENT_OF LPAREN class_expression RPAREN;

enumerated_class ::=
     ENUMERATED_CLASS LCURL object_list RCURL;

class_list ::=
     class_list COMMA class_list | class_list COMMA class_expression;

object_list ::=
     IDF | object_list COMMA IDF;

some_restriction ::=
     SOME LPAREN property_expression class_expression RPAREN ;

all_restriction ::=
```

```
     ALL LPAREN property_expression class_expression RPAREN ;

has_value_restriction ::=
     HAS_VALUE LPAREN property_expression IDF RPAREN ;

has-self_restriction ::=
     HAS_SELF LPAREN property_expression RPAREN ;

axioms ::=
     equivalent_class_axiom
     |disjoint_class_axiom
     |equivalent_property_axiom
     |disjoint_property_axiom
     |equivalent_variable_axiom
     |disjoint_variable_axiom
     |inverse_properties ;

equivalent_class_axiom ::=
     EQUIVALENT_CLASS_AXIOM LPAREN class_expression COMMA
     class_expression_list RPAREN ;

class_expression_list ::=
      class_expression | class_expression_list COMMA class_expression ;

disjoint_class_axiom ::=
     DISJOINT_CLASS_AXIOM LPAREN class_expression COMMA
     class_expression_list RPAREN ;

equivalent_property_axiom ::=
     EQUIVALENT_PROPERTY_AXIOM LPAREN
     property_expression COMMA property_expression_list RPAREN ;

property_expression_list ::=
      property_expression | property_expression_list COMMA property_expression ;

disjoint_property_axiom ::=
     DISJOINT_PROPERTY_AXIOM LPAREN property_expression COMMA
     property_expression_list RPAREN ;

equivalent_variable_axiom ::=
     EQUIVALENT_VARIABLE_AXIOM LPAREN IDF COMMA variable_list RPAREN ;

disjoint_variable_axiom ::=
     DISJOINT_VARIABLE_AXIOM LPAREN IDF COMMA variable_list RPAREN ;
```

```
variable_list ::=
      IDF | IDF COMMA IDF ;

association ::=
     ASSOCIATION IDF
     super_association_descriptor
     documentations
     association_roles
     association_variables ;

super_association_descriptor ::=
     // vide | SUPER_ASSOCIATION super_association_list ;

super_association_list ::=
     IDF | IDF COMMA super_association_list ;

association_variables ::=
     // vide | VARIABLES variables ;

association_roles ::=
     // vide | ROLES roles ;

roles ::=
     // vide | roles ROLE IDF role_descriptors ;

role_descriptors ::=
     // vide | role_descriptors role_descriptor ;

role_descriptor ::=
     TYPE IDF
     | DOMAIN domain_role
     | MULTIPLICITY role_multiplicity
     | documentations ;

role_multiplicity ::=
     min_bound max_bound ;

domain_role ::=
     LCURL instance_values RCURL ;

instance_values ::=
     instance_value | instance_values COMMA instance_value ;

instance_value ::= IDF ;
```

```
property ::=
     PROPERTY IDF
     super_property_descriptor
     documentations
     property_roles
     property_characteristics ;

property_expression ::=
     property
     |inverse_property ;

super_property_descriptor ::=
     // vide | SUPER_PROPERTY super_property_list ;

super_property_list ::=
     IDF
     | IDF COMMA super_property_list ;

property_characteristics ::=
//vide | prop_predefined_ch property_characteristics ;

prop_predefined_ch ::=
     TRANSITIVE EQUAL BOOLEAN_LITERAL
     |SYMMETRIC EQUAL BOOLEAN_LITERAL
     |FUNCTIONAL EQUAL BOOLEAN_LITERAL
     |INVERSFUNCTIONAL EQUAL BOOLEAN_LITERAL
     |SYMMETRIC EQUAL BOOLEAN_LITERAL
     |REFLEXIVE EQUAL BOOLEAN_LITERAL
     |IRREFLEXIVE EQUAL BOOLEAN_LITERAL ;

property_roles ::=
     PROPERTY_DOMAIN role_descriptors
     PROPERTY_RANGE role_descriptors;

inverse_property ::=
     INVERSE_PROPERTY LPAREN IDF RPAREN;

inverse_properties ::=
     INVERSE_PROPERTIES LPAREN property_expression,
     property_expression RPAREN;

instance ::=
     INSTANCE IDF
     ISA class_expression
     instance_descriptors;
```

```
instance_descriptors ::=
     // vide | instance_descriptors instance_descriptor ;

instance_descriptor ::=
     IDF EQUAL value | documentations;

tuple ::=
     TUPLE
     ISA IDF
     tuple_descriptors ;

tuple_descriptors ::=
     // vide | tuple_descriptors tuple_descriptor ;

tuple_descriptor ::=
     IDF EQUAL value | documentations ;

equations ::=
     // vide | equations equation ;

equation ::=
     portee COLON expression_equation | expression_equation ;

portee ::=
     portee_variable | portee COMMA portee_variable ;

portee_variable ::=
     IDF IN expr_cond ;

expression_equation ::=
     expr_gauche EQUAL expr_droite ;

expr_gauche ::=
     IDF DOT IDF | IDF ;

expr_droite ::=
     expr_cond | IF expr_cond
     THEN
     expr_droite
     ELSE
     expr_droite ;

expr_quantifiee ::=
     ALL IDF IN expr_cond COLON expr_cond
```

```
     | EXISTS IDF IN expr_cond COLON expr_cond
     | ALL IDF IN expr_cond COMMA expr_quantifiee
     | EXISTS IDF IN expr_cond COMMA expr_quantifiee ;

expr_cond ::=
     expr_quantifiee
     | expr_cond AND expr_cond
     | expr_cond OR expr_cond
     | NOT expr_cond
     | expr_cond EQUAL expr_cond
     | expr_cond INEQUAL expr_cond
     | expr_cond GT expr_cond
     | expr_cond GTE expr_cond
     | expr_cond LT expr_cond
     | expr_cond LTE expr_cond
     | expr_cond MEMBER expr_cond
     | expr_cond PLUS expr_cond
     | expr_cond MINUS expr_cond
     | expr_cond MULT expr_cond
     | expr_cond DIVISION expr_cond
     | expr_cond DIV expr_cond
     | expr_cond MOD expr_cond
     | expr_cond POWER expr_cond
     | LOG10 LPAREN expr_cond RPAREN
     | LN LPAREN expr_cond RPAREN
     | EXP LPAREN expr_cond RPAREN
     | SQRT LPAREN expr_cond RPAREN
     | SIN LPAREN expr_cond RPAREN
     | COS LPAREN expr_cond RPAREN
     | TAN LPAREN expr_cond RPAREN
     | ABS LPAREN expr_cond RPAREN
     | ROUND LPAREN expr_cond RPAREN
     | FLOOR LPAREN expr_cond RPAREN
     | CEIL LPAREN expr_cond RPAREN
     | MIN2 LPAREN expr_cond COMMA expr_cond RPAREN
     | MAX2 LPAREN expr_cond COMMA expr_cond RPAREN
     | SIZE LPAREN expr_cond RPAREN
     | LEFT LPAREN expr_cond COMMA expr_cond RPAREN
     | RIGHT LPAREN expr_cond COMMA expr_cond RPAREN
     | INDEX_OF LPAREN expr_cond COMMA expr_cond RPAREN
     | LENGTH LPAREN expr_cond RPAREN
     | SUBSTRING LPAREN expr_cond
     COMMA expr_cond COMMA expr_cond RPAREN
     | NTH LPAREN expr_cond COMMA expr_cond RPAREN
     | RANDOM LPAREN expr_cond COMMA expr_cond RPAREN
```

```
     | CONCATENATE LPAREN portee COLON expr_droite
     COMMA expr_cond RPAREN
     | MIN_ELEMENT LPAREN portee COLON expr_droite RPAREN
     | MAX_ELEMENT LPAREN portee COLON expr_droite RPAREN
     | MIN_VALUE LPAREN portee COLON expr_droite RPAREN
     | MAX_VALUE LPAREN portee COLON expr_droite RPAREN
     | SUM LPAREN portee COLON expr_droite RPAREN
     | PRODUCT LPAREN portee COLON expr_droite RPAREN
     | AVERAGE LPAREN portee COLON expr_droite RPAREN
     | STD_DEV LPAREN portee COLON expr_droite RPAREN
     | VARIANCE LPAREN portee COLON expr_droite RPAREN
     | MINUS expr_cond %prec UMINUS
     | LPAREN expr_cond RPAREN
     | expr_cond DOT IDF
     | expr_cond AT IDF BANG IDF
     | expr_cond BANG IDF
     | BOOLEAN_LITERAL
     | IDF
     | INTEGER_LITERAL
     | FLOAT_LITERAL
     | STRING_LITERAL
     | SET LPAREN portee COLON expr_droite RPAREN
     | KNOWN_SET LPAREN portee COLON expr_droite RPAREN
     | SELECT LPAREN portee COLON expr_droite RPAREN
     | UNION LPAREN portee COLON expr_droite RPAREN
     | INTER LPAREN portee COLON expr_droite RPAREN
     | LBRACK RBRACK
     | LBRACK liste_expr_cond RBRACK
     | LCURL RCURL
     | LCURL liste_intervalle RCURL
     | IS_DEFINED LPAREN expr_droite RPAREN
     | IS_VOID LPAREN expr_droite RPAREN
     | IS_KNOWN LPAREN expr_droite RPAREN
     | __ERROR LPAREN expr_droite RPAREN ;

liste_intervalle ::=
     intervalle | liste_intervalle COMMA intervalle ;

intervalle ::=
     expr_cond DOTDOT expr_cond | expr_cond ;

liste_expr_cond ::=
     expr_cond | liste_expr_cond COMMA expr_cond ;

string_literal ::=
```

```
     QUOTATION string_element QUOTATION ;

string_element ::=
     LETTRE string_element
     |DIGIT string_element
     |SPACE string_element
     |LETTRE
     |DIGIT
     |SPACE ;

integer_literal ::= DecIntegerLiteral ;

float_literal ::= DoubleLiteral ;
```

Terminaux du langage AROM :

```
LineTerminator = \r|\n|\r\n
InputCharacter = [^\r\n]
WhiteSpace = {LineTerminator} | [ \t\f]
Comment = {TraditionalComment} | {EndOfLineComment}
TraditionalComment = "/*" [^*] {CommentContent} \*+ "/"
EndOfLineComment = "//" {InputCharacter}* {LineTerminator}
CommentContent = ( [^*] | \*+[^*/] )*
Identifier = [:jletter:][:jletterdigit:]*
DecIntegerLiteral = 0 | [1-9][0-9]*
DoubleLiteral = {FLit1}|{FLit3}|{FLit4}
FLit1 = [0-9]+ \. [0-9]+ {Exponent}?
FLit3 = [0-9]+ {Exponent}
FLit4 = [0-9]+ {Exponent}?
Exponent = [eE] [+\-]? [0-9]+
StringCharacter = [^\r\n\"\\]

LETTRE = [a-z]+[A-Z]
DIGIT = [0-9]
SPACE = " "

KNOWLEDGE_BASE ::= "knowledge-base:" ;
IMPORT ::= "import:" ;
DOCUMENTATION ::= "documentation:" ;
CLASS ::= "class:" ;
SUPER_CLASS ::= "super-class:" ;
ASSOCIATION ::= "association:" ;
SUPER_ASSOCIATION ::= "super-association:" ;
PROPERTY ::= "property:" ;
SUPER_PROPERTY ::= "super-property:" ;
```

```
INVERSE_PROPERTY ::= "inverse-property" ;
INVERSE_PROPERTIES ::= "inverse-properties" ;

INTERSECTION_OF ::= "intersection-of" ;
UNION_OF ::= "union-of" ;
COMPLEMENT_OF ::= "complement-of" ;
ENUMERATED_CLASS ::= "one-of" ;
SOME ::= "some" ;
HAS_VALUE ::= "has-value" ;
HAS_SELF ::= "has-self" ;
MIN_CARDINALITY ::= "min" ;
MAX_CARDINALITY ::= "max" ;
EXACT_CARDINALITY ::= "exact" ;
SOME_VALUES ::= "some-values-in" ;
ALL_VALUES ::= "all-values-in" ;
VALUE_EQUALS ::= "value-is" ;
VAR_MIN_CARD ::= "variable-min-card" ;
VAR_MAX_CARD ::= "variable-max-card" ;
VAR_EXACT_CARD ::= "variable-exact-card" ;

EQUIVALENT_CLASS_AXIOM ::= "equivalent-classes" ;
DISJOINT_CLASS_AXIOM ::= "disjoint-classes" ;
EQUIVALENT_PROPERTY_AXIOM ::= "equivalent-properties" ;
DISJOINT_PROPERTY_AXIOM ::= "disjoint-properties" ;
EQUIVALENT_VARIABLE_AXIOM ::= "equivalent-variables" ;
DISJOINT_VARIABLE_AXIOM ::= "disjoint-variables" ;


VERSION_INFO ::= "version-info" ;
SEE_ALSO ::= "see-also" ;
IS_DEFINED_BY ::= "is-defined-by" ;
LABEL ::= "label" ;
PRIOR_VERSION ::= "prior-version" ;
INCOMPATIBLE_WITH ::= "incompatible-with" ;

TRANSITIVE ::= "transitive" ;
SYMMETRIC ::= "symmetric" ;
FUNCTIONAL ::= "functional" ;
INVERSFUNCTIONAL ::= "inverse-functional" ;
ASYMMETRIC ::= "asymmetric" ;
REFLEXIVE ::= "reflexive" ;
IRREFLEXIVE ::= "irreflexive" ;

PROPERTY_DOMAIN ::= "property-domain" ;
PROPERTY_RANGE ::= "property-range" ;
```

```
VARIABLES ::= "variables:" ;
VARIABLE ::= "variable:" ;
ROLES ::= "roles:" ;
ROLE ::= "role:" ;
INSTANCE ::= "instance:" ;
TUPLE ::= "tuple:" ;
ISA ::= "is-a:" ;
TYPE ::= "type:" ;
DOMAIN ::= "domain:" ;
DEFAULT ::= "default:" ;
DEFINITION ::= "definition:" ;
UNIT ::= "unit:" ;
ATTACHMENT ::= "attachment:" ;

SET_OF ::= "set-of" ;
LIST_OF ::= "list-of" ;

MIN ::= "min:" ;
MAX ::= "max:" ;
MULTIPLICITY ::= "multiplicity:" ;
CARDINALITY ::= "cardinality:" ;

INTEGER ::= "integer" ;
FLOAT ::= "float" ;
BOOLEAN ::= "boolean" ;
STRING ::= "string" ;
IRI ::= "IRI" ;
POINT ::= "point" ;
LINE ::= "line" ;
LINEARRING ::= "linearRing" ;
POLYGON ::= "polygon" ;
MULTILINE ::= "multiLine" ;
MULTIAREA ::= "multiArea" ;
INSTANT ::= "instant" ;
INTERVAL ::= "interval" ;
DURATION ::= "duration" ;
MULTIINTERVAL ::= "multiInterval" ;
MULTIINSTANT ::= "multiInstant" ;

BOOLEAN_LITERAL ::= "true" | "false";
QUOTATION ::= """;
LBRACK ::= "[" ;
RBRACK ::= "]" ;
LCURL ::= "{" ;
```

```
RCURL ::= "}" ;
COMMA ::= "," ;
DOTDOT ::= ".." ;
EQUAL ::= "=" ;
COLON ::= ":" ;
LTE ::= "<=" ;
GTE ::= ">=" ;
INEQUAL ::= "<>" ;
LT ::= "<" ;
GT ::= ">" ;
PLUS ::= "+" ;
MINUS ::= "-" ;
MULT ::= "*" ;
DIVISION ::= "/" ;
POWER ::= "^" ;
LPAREN ::= "(" ;
RPAREN ::= ")" ;
DOT ::= "." ;
BANG ::= "!" ;
AT ::= "@" ;
IN ::= "in" ;
MEMBER ::= "member" ;
THEN ::= "then" ;
ELSE ::= "else" ;
IF ::= "if" ;
AND ::= "and" ;
OR ::= "or" ;
NOT ::= "not" ;
LOG10 ::= "log10" ;
LN ::= "ln" ;
EXP ::= "exp" ;
SIN ::= "sin" ;
COS ::= "cos" ;
TAN ::= "tan" ;
SQRT ::= "sqrt" ;
MIN_ELEMENT ::= "minElement" ;
MAX_ELEMENT ::= "maxElement" ;
MIN_VALUE ::= "minValue" ;
MAX_VALUE ::= "maxValue" ;
MIN2 ::= "min" ;
MAX2 ::= "max" ;
SUM ::= "sum";
PRODUCT ::= "product" ;
UNION ::= "union";
INTER ::= "inter";
```

```
SIZE ::= "size" ;
SELECT ::= "select";
KNOWN_SET ::= "knownSet";
SET ::= "set";
RANDOM ::= "random";
DIV ::= "div";
MOD ::= "mod";
ABS ::= "abs";
NTH ::= "nth";
LEFT ::= "left";
RIGHT ::= "right";
LENGTH ::= "length";
INDEX_OF ::= "indexOf";
SUBSTRING ::= "substring";
CONCATENATE ::= "concatenate";
ROUND ::= "round";
CEIL ::= "ceil";
FLOOR ::= "floor";
AVERAGE ::= "average";
STD_DEV ::= "stdDev";
VARIANCE ::= "variance";
ALL ::= "all";
EXISTS ::= "exists";
IS_VOID ::= "isVoid";
IS_KNOWN ::= "isKnown";
IS_DEFINED ::= "isDefined";
```

Annexe B

Sémantique formelle de AROM-ONTO

B.1 Vocabulaire

Nous avons complété les spécifications de la sémantique dénotationnelle de AROM définie dans [49], à l'instar de la sémantique définie pour OWL 2, comme suit : Un *DatatypeMap* AROM-ONTO est le 7-uplet $D = \left(V_T, L, DI, DS, \bullet^T, \bullet^L, \bullet^{DR}\right)$ où :

- V_T représente l'ensemble de symboles de types (les noms des types reconnus par AROM-ONTO) ;
- L est une fonction qui attache à chaque type $T \in V_T$ un *espace lexical*, $L(T)$, (l'ensemble de symboles de valeurs qui peuvent être attachées au type T) ;
- pour chaque type de données, $T \in V_T$, la fonction d'interprétation $\bullet^T$ attache à T un ensemble $(T)^T$ appelé *espace de valeurs* de T ;
- DI est une fonction qui attache à chaque type $T \in V_T$ un ensemble de restrictions de domaine de la forme $DI(T) = [(v_1 |*) .. (v_2|*)]$, où $v_1, v_2 \in (T)^T, v_1 \leq v_2$ et $|DI(T)| \leq 1$;
- DS est une fonction qui attache à chaque type $T \in V_T$ un ensemble de restrictions de domaine de la forme $DS(T) = \{v_1, \ldots v_n\}$, où $\forall i \geq 1 : v_i \in (T)^T$, et $|DS(T)| \leq 1$;
- pour chaque type de données $T \in V_T$ et pour chaque *valeur lexicale*, $v \in L(T)$, la fonction d'interprétation $\bullet^L$ associe à la paire (v, T) une valeur concrète $(v, T)^L \in (T)^T$;
- pour chaque type de données $T \in V_T$ et pour chaque restriction de type intervalle *ri*, $ri \in DI(T)$, ou ensemble *re*, $re \in DS(T)$, la fonction d'interprétation $\bullet^{DR}$ associe à *ri* respectivement *re* l'ensemble $(ri)^{DR} \subseteq (T)^T$, respectivement $(re)^{DR} \subseteq (T)^T$.

La vocabulaire utilisé par une base de connaissance AROM-ONTO est défini comme étant le 8-uplet $V = (\mathcal{C}, \mathcal{X}, \mathcal{A}, \mathcal{R}, \mathcal{P}, \mathcal{DT}, \mathcal{O}, \mathcal{H}, \mathcal{LT}, \mathcal{G})$. Les composantes de V sont définies comme suit :

- $\mathcal{C}$ est l'ensemble de symboles de classe, qui contient au moins les classes `Thing` et `Nothing` ;
- $\mathcal{X}$ représente l'ensemble de symboles de variables ;
- $\mathcal{A}$ est l'ensemble de symboles d'associations ;
- $\mathcal{P}$ est l'ensemble de symboles de propriétés, et contient au moins les propriétés `TopProperty` et `BottomProperty` ;
- $\mathcal{R}$ représente l'ensemble de symboles de rôles ;
- $\mathcal{DT}$ est l'ensemble de symboles identifiant les types de données, $\mathcal{DT} \subseteq V_T$;
- $\mathcal{O}$ est l'ensemble d'objets ;
- $\mathcal{H}$ est l'ensemble de tuples ;
- $\mathcal{LT}$ est l'ensemble de valeurs concrètes ;
- $\mathcal{G}$ est l'ensemble de symboles de restriction de type. AROM-ONTO met à disposition deux possibilités pour restreindre un type de base $T \in V_T$ afin d'obtenir un sous-type de T, T', tel

que $(T')^T \subseteq (T)^T$:
- *(i)* en spécifiant l'ensemble de données acceptées à travers une restriction $G_1 = \left\{v | v \in (T)^T\right\}$ (restriction de type ensemble) et
- *(ii)* en spécifiant un sous-intervalle du type de base, à travers une restriction $G_2 = [(v_1|*)\,..\,(v_2|*)]$, ou $v_1, v_2 \in (T)^T, * \in \left\{min\left((T)^T\right), max\left((T)^T\right)\right\}$ (restriction de type intervalle, valable exclusivement pour les types ordonnés).

B.2 Interprétation

Définition B.1. *On appelle tuple sur Y étiqueté par X, une fonction qui, pour deux ensembles finis X et Y, associe à chaque élément de X un élément de Y. Un tuple étiqueté T qui associe aux éléments $x_i \in X$ des éléments $y_i \in Y, i \in 1..k$ est dénoté par $[x_1 : y_1, \ldots x_k : y_k]$. On peut également utiliser $T[x_i]$ pour désigner l'élément y_i.*

Notation B.1. *Dans un domaine abstraitΔ_I, par Λ on dénote tous les n-uplets, $n \geq 2$, qui peuvent exister entre les individus de Δ_I : $\Lambda =_{def} (\times_{i=2}^{\infty} \Delta_I)$.*

Étant donné le *DatatypeMap D* et le vocabulaire V introduits dans la section précédente, une interprétation $I = \left(\Delta_I, \Delta_D, \bullet^{\mathcal{C}}, \bullet^{\mathcal{A}}, \bullet^{\mathcal{P}}, \bullet^{\mathcal{X}}, \bullet^{\mathcal{R}}, \bullet^{\mathcal{DT}}, \bullet^{\mathcal{O}}, \bullet^{\mathcal{H}}, \bullet^{\mathcal{G}}, \bullet^{\mathcal{LT}}\right)$, d'une base de connaissances AROM-ONTO, pour D et V est définie comme suit :
- Δ_I est un ensemble non vide appelé *domaine abstrait* (ou domaine d'objets) ;
- Δ_D est un ensemble non vide, disjoint par rapport à Δ_I, appelé *domaine concret*, tel que $(T)^T \subseteq \Delta_D, \forall T \in \mathcal{DT}$;
- $\bullet^{\mathcal{C}}$ est la fonction d'interprétation des classes, qui attache à chaque classe $C \in \mathcal{C}$ un ensemble d'objets $(C)^{\mathcal{C}} \subseteq \Delta_I$ tel que :
  - $(\texttt{Thing})^{\mathcal{C}} = \Delta_I$ et
  - $(\texttt{Nothing})^{\mathcal{C}} = \emptyset$.
- $\bullet^{\mathcal{A}}$ est la fonction d'interprétation des associations, qui associe à chaque association $A \in \mathcal{A}$ un ensemble de tuples étiquetés par $\mathcal{R}$ sur Λ, $(A)^{\mathcal{A}} \subseteq 2^R \times 2^{\Lambda}$;
- $\bullet^{\mathcal{P}}$ est la fonction d'interprétation des propriétés, qui associe à chaque propriété $P \in \mathcal{P}$ un ensemble de paires, $(P)^{\mathcal{P}} \subseteq \Delta_I \times \Delta_I$ tel que :
  - $(\texttt{TopProperty})^{\mathcal{P}} = \Delta_I \times \Delta_I$ et
  - $(\texttt{BottomProperty})^{\mathcal{P}} = \emptyset$.
- $\bullet^{\mathcal{X}}$ est la fonction d'interprétation des variables, qui attache à chaque variable $x \in \mathcal{X}$ un ensemble $(x)^{\mathcal{X}} \subseteq \Delta_I \times \Delta_D$;
- $\bullet^{\mathcal{R}}$ est la fonction d'interprétation des rôles, qui attache à chaque paire rôle - association un ensemble d'individus : $(r,A)^{\mathcal{R}} \subseteq \Delta_I, \forall r \in \mathcal{R}, \forall A \subseteq \mathcal{A}$. Cette fonction est étendue pour les propriétés, de manière à prendre en compte deux rôles prédéfinis : `property-domain` et `property-range` : $(\texttt{property-domain}, P)^{\mathcal{R}} \subseteq \Delta_I$ et $(\texttt{property-range}, P)^{\mathcal{R}} \subseteq \Delta_I, \forall P \in \mathcal{P}$;
- $\bullet^{\mathcal{DT}}$ est la fonction d'interprétation des types de données, qui attache à chaque type de données $DT \in \mathcal{DT}$ un ensemble de valeurs, $(DT)^{\mathcal{DT}} \subseteq \Delta_D$, tel que $\bullet^{\mathcal{DT}}$ est équivalente à la fonction $\bullet^T$ définie pour D ;

– $\bullet^{O}$ est la fonction d'interprétation d'individus, qui attache à chaque individu $a \in O$, un élément de Δ_I : $(a)^{O} \in \Delta_I$;
– $\bullet^{\mathcal{H}}$ est la fonction d'interprétation des tuples, qui attache à chaque tuple $t \in \mathcal{H}$, un élément de $(t)^{\mathcal{H}} \in 2^{\mathcal{R}} \times 2^{\Delta_I}$. De plus, si $\mathtt{A} \in \mathcal{A}$ est l'association à laquelle $\mathtt{t}$ est attaché, $(\mathtt{t})^{\mathcal{H}} \in (\mathtt{A})^{\mathcal{A}}$, et si le nombre de rôles définis par $\mathtt{A}$ est n, alors $(\mathtt{t!r}_i)^{O} \in (\mathtt{r}_i, \mathtt{A})^{\mathcal{R}}, \forall i \in 1..n$. Cette fonction est spécialisée pour les tuples des propriétés comme suit : $\forall p \in \mathcal{H}, (p)^{\mathcal{H}} \in (P)^{\mathcal{P}}$.
– $\bullet^{\mathcal{G}}$ est la fonction d'interprétation des restrictions de type, définie comme suit : $\forall r \in \mathcal{G}, \forall T \in V_T : (T,r)^{\mathcal{G}} \subseteq (T)^{T}$;
– $\bullet^{\mathcal{LT}}$ est la fonction d'interprétation des valeurs concrètes, définie comme suit : $(lt)^{\mathcal{LT}} = (LV,T)^{L}, \forall lt \in \mathcal{LT}$, où LV est la forme lexicale de lt, $LV \in L(T)$ et T est le type de lt.

Étant donné le vocabulaire V, les conventions suivantes sont utilisées dans ce document afin de différencier parmi les éléments syntaxiques utilisés :

– `P` est une propriété, $\mathtt{P} \in \mathcal{P}$;
– `PE` est une expression à base de propriétés ;
– `C` est une classe, $\mathtt{C} \in \mathcal{C}$;
– `CE` est une expression à base de classes ;
– `o` est un objet, $\mathtt{o} \in O$;
– `t` est un tuple, $\mathtt{t} \in \mathcal{H}$;
– `D` est un domaine, ...
– `lt` est une valeur concrète.

Pour toute classe $\mathtt{C} \in \mathcal{C}$, par `C.v` on dénote la variable $\mathtt{v}, \mathtt{v} \in \mathcal{X}$ telle que définie ou héritée par la classe `C`. Pour toute propriété $\mathtt{P} \in \mathcal{P}$, par `P.v` on dénote une des caractéristiques prédéfinies des propriétés AROM-ONTO : v∈{`transitive`, `symmetric`, `functional`, `invers-functional`, `asymmetric`, `reflexive`, `irreflexive`}. De plus, par `P!r` on dénote un rôle `r`, $\mathtt{r} \in \{\mathtt{property\text{-}domain}, \mathtt{property\text{-}range}\}$. Pour une association $\mathtt{A} \in \mathcal{A}$, par `A.v` on dénote une variable `v` telle que définie ou héritée par l'association `A` et par `A!r` on dénote un rôle de l'association `A`.

B.2.1 Interprétation des expressions à base de propriétés

La fonction d'interprétation des propriétés est étendue pour prendre en compte les expressions à base de propriétés :

Expression à base de propriétés	Interprétation $\bullet^{\mathcal{P}}$
`inverse-property (P)`, $\mathtt{P} \in \mathcal{P}$	$\left\{(x,y) \mid (y,x) \in (\mathtt{P})^{\mathcal{P}}, x,y \in \Delta_I\right\}$

TAB. *B.1: L'interprétation des expressions à base de propriétés de AROM-ONTO.*

B.2.2 Interprétation des domaines

La fonction d'interprétation des types de données est étendue pour prendre en compte les domaines AROM-ONTO, comme illustré dans le tableau B.2.

Domaine	Interprétation $\bullet^{\mathcal{DT}}$
`type:T domain : [val1..val2]`	$\left[(val_1)^{\mathcal{LT}} ..(val_2)^{\mathcal{LT}}\right] \subseteq (\mathtt{T})^T$, où $(\mathtt{val}_1)^{\mathcal{LT}}, (\mathtt{val}_2)^{\mathcal{LT}} \in (\mathtt{T})^T$
`type :T domain : {val1...valn}`	$\left\{(val_1)^{\mathcal{LT}}, \ldots, (val_n)^{\mathcal{LT}}\right\}, i \in 1..n$, où $(\mathtt{val}_i)^{\mathcal{LT}} \in (\mathtt{T})^T, \forall i \in 1..n$
`type-restriction : T,r1,...,rn`	$(T)^T \bigcap (T,\mathtt{r}_1)^{\mathcal{G}} \bigcap \ldots \bigcap (T,\mathtt{r}_n)^{\mathcal{G}}$ où $\mathtt{r}_i \in \mathcal{G}, \forall i \in 1..n$

TAB. *B.2: L'interprétation des domaines définis en AROM-ONTO.*

B.2.3 Interprétation des expressions à base de classes

La fonction d'interprétation des classes est étendue pour les expressions à base de classes comme le montre le tableau B.3.

Expression à base de classes	Interprétation $\bullet^{\mathcal{C}}$
`intersection(CE1,...,CEn)`	$(\mathtt{CE}_1)^{\mathcal{C}} \bigcap \ldots \bigcap (\mathtt{CE}_n)^{\mathcal{C}}$
`union(CE1,...,CEn)`	$(\mathtt{CE}_1)^{\mathcal{C}} \bigcup \ldots \bigcup (\mathtt{CE}_n)^{\mathcal{C}}$
`complement(CE)`	$\Delta_I \setminus (CE)^{\mathcal{C}}$
`one-of{o1,...on}` , $\mathtt{o}_i \in O \forall i \in 1..n$	$\left\{(o_1)^O, \ldots (o_n)^O\right\}$
`some ( PE CE )`	$\left\{o \middle\vert \exists x : \langle o,x\rangle \in (PE)^{\mathcal{P}} \wedge x \in (CE)^{\mathcal{C}}\right\}$
`all ( PE CE )`	$\left\{o \middle\vert \forall x : \langle o,x\rangle \in (PE)^{\mathcal{P}} \rightarrow x \in (CE)^{\mathcal{C}}\right\}$
`self( PE )`	$\left\{o \middle\vert \langle o,o\rangle \in (PE)^{\mathcal{P}}\right\}$
`has-value ( PE x)`	$\left\{o \middle\vert \left\langle o,(x)^O\right\rangle \in (PE)^{\mathcal{P}}\right\}$
`min ( n PE CE )`	$\left\{o \middle\vert card\left(\left\{x \middle\vert (o,x) \in (PE)^{\mathcal{P}} \wedge x \in (CE)^{\mathcal{C}}\right\}\right) \geq n\right\}$
`max ( n PE CE)`	$\left\{o \middle\vert card\left(\left\{x \middle\vert (o,x) \in (PE)^{\mathcal{P}} \wedge x \in (CE)^{\mathcal{C}}\right\}\right) \leq n\right\}$
`exact( n PE CE)`	$\left\{o \middle\vert card\left(\left\{x \middle\vert (o,x) \in (PE)^{\mathcal{P}} \wedge x \in (CE)^{\mathcal{C}}\right\}\right) = n\right\}$
`min ( n PE )`	$\left\{o \middle\vert card\left(\left\{x \middle\vert (o,x) \in (PE)^{\mathcal{P}}\right\}\right) \geq n\right\}$
`max ( n PE )`	$\left\{o \middle\vert card\left(\left\{x \middle\vert (o,x) \in (PE)^{\mathcal{P}}\right\}\right) \leq n\right\}$
`exact ( n PE )`	$\left\{o \middle\vert card\left(\left\{x \middle\vert (o,x) \in (PE)^{\mathcal{P}}\right\}\right) = n\right\}$
`some-values-in(v1,...,vn D)`, $n \geq 1$	$\left\{o \middle\vert \exists y_1, \ldots, y_n : (o,y_i) \in (v_i)^{\mathcal{X}} \wedge y_i \in (D)^{\mathcal{DT}}, \forall i \in 1..n\right\}$
`all-values-in(v1,...,vn D)`, $n \geq 1$	$\left\{o \middle\vert \forall y_1, \ldots, y_n : (o,y_i) \in (v_i)^{\mathcal{X}}\ \forall i \in 1..n \rightarrow y_i \in (D)^{\mathcal{DT}}\right\}$
`value-is (v lt)`	$\left\{o \middle\vert \left(o,(lt)^{\mathcal{LT}}\right) \in (v)^{\mathcal{X}}\right\}$
`variable-min-card (v n)`, $n \geq 0$	$\left\{o \middle\vert card\left(\left\{x \middle\vert (o,x) \in (v)^{\mathcal{X}}\right\}\right) \geq n\right\}$
`variable-max-card (v n)`, $n \geq 0$	$\left\{o \middle\vert card\left(\left\{x \middle\vert (o,x) \in (v)^{\mathcal{X}}\right\}\right) \leq n\right\}$
`variable-exact-card (v n)`, $n \geq 0$	$\left\{o \middle\vert card\left(\left\{x \middle\vert (o,x) \in (v)^{\mathcal{X}}\right\}\right) = n\right\}$
`variable-min-card (v n D)`, $n \geq 0$	$\left\{o \middle\vert card\left(\left\{x \middle\vert (o,x) \in (v)^{\mathcal{X}} \wedge x \in (D)^{\mathcal{DT}}\right\}\right) \geq n\right\}$
`variable-max-card (v n D)`, $n \geq 0$	$\left\{o \middle\vert card\left(\left\{x \middle\vert (o,x) \in (v)^{\mathcal{X}} \wedge x \in (D)^{\mathcal{DT}}\right\}\right) \leq n\right\}$
`variable-exact-card (v n D)`, $n \geq 0$	$\left\{o \middle\vert card\left(\left\{x \middle\vert (o,x) \in (v)^{\mathcal{X}} \wedge x \in (D)^{\mathcal{DT}}\right\}\right) = n\right\}$

TAB. *B.3: L'interprétation des expressions à base de classes de AROM-ONTO*

B.2.4 Axiomes

Les tableaux B.4, B.5 et B.6 présentent les conditions qui doivent êtres remplies par une base de connaissances O pour que les axiomes qu'elle contient soit satisfaits dans une interprétation I.

Axiome	Condition
$\texttt{CE}_1\texttt{.super-class: CE}_2, \texttt{CE}_3, \ldots, \texttt{CE}_n$	$(CE_1)^C \subseteq (CE_2)^C \wedge (CE_1)^C \subseteq (CE_3)^C \wedge \ldots \wedge (CE_1)^C \subseteq (CE_n)^C$
$\texttt{equivalent-classes(CE}_1, \ldots, \texttt{CE}_n\texttt{)}$	$(CE_i)^C = (CE_j)^C, \forall i \in 1..n, \forall j \in 1..n$
$\texttt{disjoint-classes(CE}_1, \ldots, \texttt{CE}_n\texttt{)}$	$(CE_i)^C \bigcap (CE_j)^C = \emptyset, \forall i \in 1..n, \forall j \in 1..n$
Thing	$(\texttt{Thing})^C = \Delta_I$
Nothing	$(\texttt{Nothing})^C = \emptyset$

TAB. *B.4: L'interprétation des axiomes à base de classes de AROM-ONTO.*

Axiome	Condition
$\texttt{P}_1\texttt{.super-property P}_2$	$(P_1)^{\mathcal{P}} \subseteq (P_2)^{\mathcal{P}}$
$\texttt{equivalent-properties(PE}_1, \ldots, \texttt{PE}_n\texttt{)}$	$(PE_i)^{\mathcal{P}} = (PE_j)^{\mathcal{P}}, \forall i \in 1..n, \forall j \in 1..n$
$\texttt{disjoint-properties(PE}_1, \ldots, \texttt{PE}_n\texttt{)}$	$(PE_i)^{\mathcal{P}} \bigcap (PE_j)^{\mathcal{P}} = \emptyset, \forall i \in 1..n, \forall j \in 1..n$
$\texttt{PE.transitive} = \texttt{true}$	$\forall x,y,z : (x,y) \in (PE)^{\mathcal{P}} \wedge (y,z) \in (PE)^{\mathcal{P}} \rightarrow (x,z) \in (PE)^{\mathcal{P}}$
$\texttt{PE.symmetric} = \texttt{true}$	$\forall x,y : (x,y) \in (PE)^{\mathcal{P}} \rightarrow (y,x) \in (PE)^{\mathcal{P}}$
$\texttt{PE.asymetric} = \texttt{true}$	$\forall x,y : (x,y) \in (PE)^{\mathcal{P}} \rightarrow (y,x) \notin (PE)^{\mathcal{P}}$
$\texttt{PE.reflexive} = \texttt{true}$	$\forall x : x \in \Delta_I \rightarrow (x,x) \in (PE)^{\mathcal{P}}$
$\texttt{PE.irreflexive} = \texttt{true}$	$\forall x : x \in \Delta_I \rightarrow (x,x) \notin (PE)^{\mathcal{P}}$
$\texttt{PE.functional} = \texttt{true}$	$\forall x,y_1,y_2 : (x,y_1) \in (PE)^{\mathcal{P}} \wedge (x,y_2) \in (PE)^{\mathcal{P}} \rightarrow y_1 = y_2$
$\texttt{PE.invers-functional} = \texttt{true}$	$\forall x_1,x_2,y : (x_1,y) \in (PE)^{\mathcal{P}} \wedge (x_2,y) \in (PE)^{\mathcal{P}} \rightarrow x_1 = x_2$
PE!property-domain CE	$\forall x,y : (x,y) \in (PE)^{\mathcal{P}} \rightarrow x \in (CE)^C$
PE!property-range CE	$\forall x,y : (x,y) \in (PE)^{\mathcal{P}} \rightarrow y \in (CE)^C$
$\texttt{inverse-properties (PE}_1\texttt{,PE}_2\texttt{)}$	$(PE_1)^{\mathcal{P}} = \left\{ (x,y) \,\middle|\, (y,x) \in (PE_2)^{\mathcal{P}} \right\}$
TopProperty	$(\texttt{TopProperty})^{\mathcal{P}} = \Delta_I \times \Delta_I$
BottomProperty	$(\texttt{BottomProperty})^{\mathcal{P}} = \emptyset$

TAB. *B.5: L'interprétation des axiomes à base de propriétés de AROM-ONTO.*

Axiome	Condition
$\texttt{equivalent-variables(v}_1, \ldots, \texttt{v}_n\texttt{)}$	$(v_i)^X = (v_j)^X, \forall i \in 1..n, \forall j \in 1..n$
$\texttt{disjoint-variables(v}_1, \ldots, \texttt{v}_n\texttt{)}$	$(v_i)^X \bigcap (v_j)^X = \emptyset, \forall i \in 1..n, \forall j \in 1..n$

TAB. *B.6: L'interprétation des axiomes à base de variables de AROM-ONTO.*

Assertion	Condition
`o is-a C`	$(o)^{\mathcal{O}} \in (C)^{\mathcal{C}}$
`t is-a A`	$(t)^{\mathcal{H}} \in (A)^{\mathcal{A}}$
`p is-a P`	$(p)^{\mathcal{H}} \in (P)^{\mathcal{P}}$
`class: C variable: v type: T`	$(v)^{\mathcal{X}} \in (C)^{\mathcal{C}} \times (T)^{\mathcal{DT}}$
`t is-a AROMSameIndividual`, où $\mathtt{t} \in \mathcal{H}, \mathtt{t} = [\mathtt{ind}_1 : \mathtt{o}_1, \mathtt{ind}_2 : \mathtt{o}_2]$	$(t)^{\mathcal{A}} \in (\mathtt{AROMSameIndividual})^{\mathcal{A}} \wedge (o_1)^{\mathcal{O}} = (o_2)^{\mathcal{O}}$
`t is-a AROMDifferentIndividuals`, où $\mathtt{t} \in \mathcal{H}, \mathtt{t} = [\mathtt{ind}_1 : \mathtt{o}_1, \mathtt{ind}_2 : \mathtt{o}_2]$	$(t)^{\mathcal{A}} \in (\mathtt{AROMDifferentIndividuals})^{\mathcal{A}} \wedge (o_1)^{\mathcal{O}} \neq (o_2)^{\mathcal{O}}$

TAB. *B.7: L'interprétation des tuples des associations prédéfinies SameIndividual et DifferentIndividuals.*

Annexe C

L'ontologie *QualitativeSpatialRelations*

```
<?xml version="1.0"?>


<!DOCTYPE Ontology [
    <!ENTITY qsr "qsr#" >
    <!ENTITY East10 "qsr#East:" >
    <!ENTITY West4 "qsr#West:" >
    <!ENTITY South7 "qsr#South:" >
    <!ENTITY North2 "qsr#North:" >
    <!ENTITY East15 "qsr#West:East:" >
    <!ENTITY West3 "qsr#North:West:" >
    <!ENTITY East9 "qsr#North:East:" >
    <!ENTITY South6 "qsr#West:South:" >
    <!ENTITY East14 "qsr#South:East:" >
    <!ENTITY South8 "qsr#North:South:" >
    <!ENTITY East11 "qsr#North:West:East:" >
    <!ENTITY East12 "qsr#West:South:East:" >
    <!ENTITY East8 "qsr#North:South:East:" >
    <!ENTITY South5 "qsr#North:West:South:" >
    <!ENTITY East13 "qsr#North:West:South:East:" >
    <!ENTITY qsr2 "http://www.w3.org/2006/12/qsr#" >
    <!ENTITY owl "http://www.w3.org/2002/07/owl#" >
    <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >
    <!ENTITY owl2xml "http://www.w3.org/2006/12/owl2-xml#" >
    <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#" >
    <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#" >
    <!ENTITY qsr "http://www.semanticweb.org/ontologies/qsr.owl#" >
    <!ENTITY East4 "http://www.semanticweb.org/ontologies/qsr.owl#East:" >
    <!ENTITY West "http://www.semanticweb.org/ontologies/qsr.owl#West:" >
    <!ENTITY South "http://www.semanticweb.org/ontologies/qsr.owl#South:" >
    <!ENTITY North "http://www.semanticweb.org/ontologies/qsr.owl#North:" >
    <!ENTITY East "http://www.semanticweb.org/ontologies/qsr.owl#West:East:" >
    <!ENTITY East2 "http://www.semanticweb.org/ontologies/qsr.owl#South:East:" >
    <!ENTITY East3 "http://www.semanticweb.org/ontologies/qsr.owl#North:East:" >
    <!ENTITY West2 "http://www.semanticweb.org/ontologies/qsr.owl#North:West:" >
    <!ENTITY South2 "http://www.semanticweb.org/ontologies/qsr.owl#West:South:" >
    <!ENTITY South3 "http://www.semanticweb.org/ontologies/qsr.owl#North:South:" >
    <!ENTITY East7 "http://www.semanticweb.org/ontologies/qsr.owl#West:South:East:" >
    <!ENTITY East6 "http://www.semanticweb.org/ontologies/qsr.owl#North:South:East:" >
    <!ENTITY South4 "http://www.semanticweb.org/ontologies/qsr.owl#North:West:South:" >
    <!ENTITY East5 "http://www.semanticweb.org/ontologies/qsr.owl#North:West:South:East:" >
]>


<Ontology xmlns="http://www.w3.org/2006/12/owl2-xml#"
     xml:base="http://www.w3.org/2006/12/owl2-xml#"
```

```
 xmlns:East4="&qsr;East:"
 xmlns:East3="&qsr;North:East:"
 xmlns:North2="qsr#North:"
 xmlns:East2="&qsr;South:East:"
 xmlns:East="&qsr;West:East:"
 xmlns:West="&qsr;West:"
 xmlns:East9="&North2;East:"
 xmlns:South="&qsr;South:"
 xmlns:East8="&North2;South:East:"
 xmlns:East7="&qsr;West:South:East:"
 xmlns:East6="&qsr;North:South:East:"
 xmlns:qsr2="http://www.w3.org/2006/12/qsr#"
 xmlns:qsr="qsr#"
 xmlns:East5="&qsr;North:West:South:East:"
 xmlns:East10="qsr#East:"
 xmlns:East11="&North2;West:East:"
 xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
 xmlns:East12="&West4;South:East:"
 xmlns:owl2xml="http://www.w3.org/2006/12/owl2-xml#"
 xmlns:East13="&North2;West:South:East:"
 xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
 xmlns:North="&qsr;North:"
 xmlns:qsr="http://www.semanticweb.org/ontologies/qsr.owl#"
 xmlns:East14="&South7;East:"
 xmlns:South7="qsr#South:"
 xmlns:East15="&West4;East:"
 xmlns:South8="&North2;South:"
 xmlns:South5="&North2;West:South:"
 xmlns:South6="&West4;South:"
 xmlns:West2="&qsr;North:West:"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
 xmlns:owl="http://www.w3.org/2002/07/owl#"
 xmlns:South3="&qsr;North:South:"
 xmlns:South4="&qsr;North:West:South:"
 xmlns:West4="qsr#West:"
 xmlns:South2="&qsr;West:South:"
 xmlns:West3="&North2;West:"
 URI="qsr">
<SubClassOf>
    <Class URI="qsr#East"/>
    <Class URI="&qsr;DirectionRelation"/>
</SubClassOf>
<Declaration>
    <Class URI="qsr#East"/>
</Declaration>
<SubClassOf>
    <Class URI="&East10;Neutral"/>
    <Class URI="&qsr;DirectionRelation"/>
</SubClassOf>
<Declaration>
    <Class URI="&East10;Neutral"/>
</Declaration>
<SubClassOf>
    <Class URI="qsr#Neutral"/>
    <Class URI="&qsr;DirectionRelation"/>
```

```
</SubClassOf>
<Declaration>
    <Class URI="qsr#Neutral"/>
</Declaration>
<SubClassOf>
    <Class URI="qsr#North"/>
    <Class URI="&qsr;DirectionRelation"/>
</SubClassOf>
<Declaration>
    <Class URI="qsr#North"/>
</Declaration>
<SubClassOf>
    <Class URI="&North2;East"/>
    <Class URI="&qsr;DirectionRelation"/>
</SubClassOf>
<Declaration>
    <Class URI="&North2;East"/>
</Declaration>
<SubClassOf>
    <Class URI="&North2;East:Neutral"/>
    <Class URI="&qsr;DirectionRelation"/>
</SubClassOf>
<Declaration>
    <Class URI="&North2;East:Neutral"/>
</Declaration>
<SubClassOf>
    <Class URI="&North2;Neutral"/>
    <Class URI="&qsr;DirectionRelation"/>
</SubClassOf>
<Declaration>
    <Class URI="&North2;Neutral"/>
</Declaration>
<SubClassOf>
    <Class URI="&North2;South"/>
    <Class URI="&qsr;DirectionRelation"/>
</SubClassOf>
<Declaration>
    <Class URI="&North2;South"/>
</Declaration>
<SubClassOf>
    <Class URI="&North2;South:East"/>
    <Class URI="&qsr;DirectionRelation"/>
</SubClassOf>
<Declaration>
    <Class URI="&North2;South:East"/>
</Declaration>
<SubClassOf>
    <Class URI="&North2;South:East:Neutral"/>
    <Class URI="&qsr;DirectionRelation"/>
</SubClassOf>
<Declaration>
    <Class URI="&North2;South:East:Neutral"/>
</Declaration>
<SubClassOf>
    <Class URI="&North2;South:Neutral"/>
```

```
    <Class URI="&qsr;DirectionRelation"/>
</SubClassOf>
<Declaration>
    <Class URI="&North2;South:Neutral"/>
</Declaration>
<SubClassOf>
    <Class URI="&North2;West"/>
    <Class URI="&qsr;DirectionRelation"/>
</SubClassOf>
<Declaration>
    <Class URI="&North2;West"/>
</Declaration>
<SubClassOf>
    <Class URI="&North2;West:East"/>
    <Class URI="&qsr;DirectionRelation"/>
</SubClassOf>
<Declaration>
    <Class URI="&North2;West:East"/>
</Declaration>
<SubClassOf>
    <Class URI="&North2;West:East:Neutral"/>
    <Class URI="&qsr;DirectionRelation"/>
</SubClassOf>
<Declaration>
    <Class URI="&North2;West:East:Neutral"/>
</Declaration>
<SubClassOf>
    <Class URI="&North2;West:Neutral"/>
    <Class URI="&qsr;DirectionRelation"/>
</SubClassOf>
<Declaration>
    <Class URI="&North2;West:Neutral"/>
</Declaration>
<SubClassOf>
    <Class URI="&North2;West:South"/>
    <Class URI="&qsr;DirectionRelation"/>
</SubClassOf>
<Declaration>
    <Class URI="&North2;West:South"/>
</Declaration>
<SubClassOf>
    <Class URI="&North2;West:South:East"/>
    <Class URI="&qsr;DirectionRelation"/>
</SubClassOf>
<Declaration>
    <Class URI="&North2;West:South:East"/>
</Declaration>
<SubClassOf>
    <Class URI="&North2;West:South:East:Neutral"/>
    <Class URI="&qsr;DirectionRelation"/>
</SubClassOf>
<Declaration>
    <Class URI="&North2;West:South:East:Neutral"/>
</Declaration>
<SubClassOf>
```

```
    <Class URI="qsr#South"/>
    <Class URI="&qsr;DirectionRelation"/>
</SubClassOf>
<Declaration>
    <Class URI="qsr#South"/>
</Declaration>
<SubClassOf>
    <Class URI="&South7;East"/>
    <Class URI="&qsr;DirectionRelation"/>
</SubClassOf>
<Declaration>
    <Class URI="&South7;East"/>
</Declaration>
<SubClassOf>
    <Class URI="&South7;East:Neutral"/>
    <Class URI="&qsr;DirectionRelation"/>
</SubClassOf>
<Declaration>
    <Class URI="&South7;East:Neutral"/>
</Declaration>
<SubClassOf>
    <Class URI="&South7;Neutral"/>
    <Class URI="&qsr;DirectionRelation"/>
</SubClassOf>
<Declaration>
    <Class URI="&South7;Neutral"/>
</Declaration>
<SubClassOf>
    <Class URI="qsr#West"/>
    <Class URI="&qsr;DirectionRelation"/>
</SubClassOf>
<Declaration>
    <Class URI="qsr#West"/>
</Declaration>
<SubClassOf>
    <Class URI="&West4;East"/>
    <Class URI="&qsr;DirectionRelation"/>
</SubClassOf>
<Declaration>
    <Class URI="&West4;East"/>
</Declaration>
<SubClassOf>
    <Class URI="&West4;East:Neutral"/>
    <Class URI="&qsr;DirectionRelation"/>
</SubClassOf>
<Declaration>
    <Class URI="&West4;East:Neutral"/>
</Declaration>
<SubClassOf>
    <Class URI="&West4;Neutral"/>
    <Class URI="&qsr;DirectionRelation"/>
</SubClassOf>
<Declaration>
    <Class URI="&West4;Neutral"/>
</Declaration>
```

```
<SubClassOf>
    <Class URI="&West4;South"/>
    <Class URI="&qsr;DirectionRelation"/>
</SubClassOf>
<Declaration>
    <Class URI="&West4;South"/>
</Declaration>
<SubClassOf>
    <Class URI="&West4;South:East"/>
    <Class URI="&qsr;DirectionRelation"/>
</SubClassOf>
<Declaration>
    <Class URI="&West4;South:East"/>
</Declaration>
<SubClassOf>
    <Class URI="&West4;South:East:Neutral"/>
    <Class URI="&qsr;DirectionRelation"/>
</SubClassOf>
<Declaration>
    <Class URI="&West4;South:East:Neutral"/>
</Declaration>
<SubClassOf>
    <Class URI="&West4;South:Neutral"/>
    <Class URI="&qsr;DirectionRelation"/>
</SubClassOf>
<Declaration>
    <Class URI="&West4;South:Neutral"/>
</Declaration>
<EquivalentClasses>
    <Class URI="&qsr;DirectionRelation"/>
    <ObjectExactCardinality cardinality="1">
        <ObjectProperty URI="&qsr;object"/>
        <Class URI="&qsr;SpatialObject"/>
    </ObjectExactCardinality>
</EquivalentClasses>
<EquivalentClasses>
    <Class URI="&qsr;DirectionRelation"/>
    <ObjectExactCardinality cardinality="1">
        <ObjectProperty URI="&qsr;subject"/>
        <Class URI="&qsr;SpatialObject"/>
    </ObjectExactCardinality>
</EquivalentClasses>
<SubClassOf>
    <Class URI="&qsr;DirectionRelation"/>
    <Class URI="&owl;Thing"/>
</SubClassOf>
<Declaration>
    <Class URI="&qsr;DirectionRelation"/>
</Declaration>
<EquivalentClasses>
    <Class URI="&qsr;DistanceRelation"/>
    <ObjectExactCardinality cardinality="1">
        <ObjectProperty URI="&qsr;hasMetric"/>
        <Class URI="&qsr;Metric"/>
    </ObjectExactCardinality>
```

```
</EquivalentClasses>
<EquivalentClasses>
    <Class URI="&qsr;DistanceRelation"/>
    <ObjectExactCardinality cardinality="1">
        <ObjectProperty URI="&qsr;hasScale"/>
        <Class URI="&qsr;Scale"/>
    </ObjectExactCardinality>
</EquivalentClasses>
<EquivalentClasses>
    <Class URI="&qsr;DistanceRelation"/>
    <ObjectExactCardinality cardinality="1">
        <ObjectProperty URI="&qsr;object"/>
        <Class URI="&qsr;SpatialObject"/>
    </ObjectExactCardinality>
</EquivalentClasses>
<EquivalentClasses>
    <Class URI="&qsr;DistanceRelation"/>
    <ObjectExactCardinality cardinality="1">
        <ObjectProperty URI="&qsr;subject"/>
        <Class URI="&qsr;SpatialObject"/>
    </ObjectExactCardinality>
</EquivalentClasses>
<Declaration>
    <Class URI="&qsr;DistanceRelation"/>
</Declaration>
<SubClassOf>
    <Class URI="&qsr;LargeScale"/>
    <Class URI="&qsr;Scale"/>
</SubClassOf>
<Declaration>
    <Class URI="&qsr;LargeScale"/>
</Declaration>
<SubClassOf>
    <Class URI="&qsr;MediumScale"/>
    <Class URI="&qsr;Scale"/>
</SubClassOf>
<Declaration>
    <Class URI="&qsr;MediumScale"/>
</Declaration>
<Declaration>
    <Class URI="&qsr;Metric"/>
</Declaration>
<Declaration>
    <Class URI="&qsr;Scale"/>
</Declaration>
<SubClassOf>
    <Class URI="&qsr;SmallScale"/>
    <Class URI="&qsr;Scale"/>
</SubClassOf>
<Declaration>
    <Class URI="&qsr;SmallScale"/>
</Declaration>
<Declaration>
    <Class URI="&qsr;SpatialObject"/>
</Declaration>
```

```
<SubClassOf>
    <Class URI="&qsr2;Close"/>
    <Class URI="&qsr;DistanceRelation"/>
</SubClassOf>
<DisjointClasses>
    <Class URI="&qsr2;Close"/>
    <Class URI="&qsr2;Far"/>
    <Class URI="&qsr2;VeryClose"/>
</DisjointClasses>
<EntityAnnotation>
    <Class URI="&qsr2;Close"/>
    <Annotation annotationURI="&rdfs;label">
        <Constant>Close</Constant>
    </Annotation>
</EntityAnnotation>
<Declaration>
    <Class URI="&qsr2;Close"/>
</Declaration>
<SubClassOf>
    <Class URI="&qsr2;Far"/>
    <Class URI="&qsr;DistanceRelation"/>
</SubClassOf>
<EntityAnnotation>
    <Class URI="&qsr2;Far"/>
    <Annotation annotationURI="&rdfs;label">
        <Constant>Far</Constant>
    </Annotation>
</EntityAnnotation>
<Declaration>
    <Class URI="&qsr2;Far"/>
</Declaration>
<SubClassOf>
    <Class URI="&qsr2;VeryClose"/>
    <Class URI="&qsr;DistanceRelation"/>
</SubClassOf>
<EntityAnnotation>
    <Class URI="&qsr2;VeryClose"/>
    <Annotation annotationURI="&rdfs;label">
        <Constant>VeryClose</Constant>
    </Annotation>
</EntityAnnotation>
<Declaration>
    <Class URI="&qsr2;VeryClose"/>
</Declaration>
<SubObjectPropertyOf>
    <ObjectProperty URI="&qsr;contains"/>
    <ObjectProperty URI="&qsr;spatiallyRelated"/>
</SubObjectPropertyOf>
<InverseObjectProperties>
    <ObjectProperty URI="&qsr;contains"/>
    <ObjectProperty URI="&qsr;inside"/>
</InverseObjectProperties>
<TransitiveObjectProperty>
    <ObjectProperty URI="&qsr;contains"/>
</TransitiveObjectProperty>
```

```
<DisjointObjectProperties>
    <ObjectProperty URI="&qsr;contains"/>
    <ObjectProperty URI="&qsr;disjoint"/>
    <ObjectProperty URI="&qsr;equals"/>
    <ObjectProperty URI="&qsr;inside"/>
    <ObjectProperty URI="&qsr;overlap"/>
    <ObjectProperty URI="&qsr;tangent_contains"/>
    <ObjectProperty URI="&qsr;tangent_inside"/>
    <ObjectProperty URI="&qsr;touches"/>
</DisjointObjectProperties>
<Declaration>
    <ObjectProperty URI="&qsr;contains"/>
</Declaration>
<ObjectPropertyDomain>
    <ObjectProperty URI="&qsr;definedAs"/>
    <Class URI="&qsr;DirectionRelation"/>
</ObjectPropertyDomain>
<Declaration>
    <ObjectProperty URI="&qsr;definedAs"/>
</Declaration>
<SubObjectPropertyOf>
    <ObjectProperty URI="&qsr;disjoint"/>
    <ObjectProperty URI="&qsr;spatiallyRelated"/>
</SubObjectPropertyOf>
<InverseObjectProperties>
    <ObjectProperty URI="&qsr;disjoint"/>
    <ObjectProperty URI="&qsr;disjoint"/>
</InverseObjectProperties>
<Declaration>
    <ObjectProperty URI="&qsr;disjoint"/>
</Declaration>
<SubObjectPropertyOf>
    <ObjectProperty URI="&qsr;equals"/>
    <ObjectProperty URI="&qsr;spatiallyRelated"/>
</SubObjectPropertyOf>
<InverseObjectProperties>
    <ObjectProperty URI="&qsr;equals"/>
    <ObjectProperty URI="&qsr;equals"/>
</InverseObjectProperties>
<SymmetricObjectProperty>
    <ObjectProperty URI="&qsr;equals"/>
</SymmetricObjectProperty>
<TransitiveObjectProperty>
    <ObjectProperty URI="&qsr;equals"/>
</TransitiveObjectProperty>
<Declaration>
    <ObjectProperty URI="&qsr;equals"/>
</Declaration>
<ObjectPropertyDomain>
    <ObjectProperty URI="&qsr;hasMetric"/>
    <Class URI="&qsr;DistanceRelation"/>
</ObjectPropertyDomain>
<ObjectPropertyRange>
    <ObjectProperty URI="&qsr;hasMetric"/>
    <Class URI="&qsr;Metric"/>
```

```
</ObjectPropertyRange>
<Declaration>
    <ObjectProperty URI="&qsr;hasMetric"/>
</Declaration>
<ObjectPropertyDomain>
    <ObjectProperty URI="&qsr;hasScale"/>
    <Class URI="&qsr;DistanceRelation"/>
</ObjectPropertyDomain>
<ObjectPropertyRange>
    <ObjectProperty URI="&qsr;hasScale"/>
    <Class URI="&qsr;Scale"/>
</ObjectPropertyRange>
<Declaration>
    <ObjectProperty URI="&qsr;hasScale"/>
</Declaration>
<SubObjectPropertyOf>
    <ObjectProperty URI="&qsr;inside"/>
    <ObjectProperty URI="&qsr;spatiallyRelated"/>
</SubObjectPropertyOf>
<TransitiveObjectProperty>
    <ObjectProperty URI="&qsr;inside"/>
</TransitiveObjectProperty>
<Declaration>
    <ObjectProperty URI="&qsr;inside"/>
</Declaration>
<ObjectPropertyDomain>
    <ObjectProperty URI="&qsr;object"/>
    <Class URI="&qsr;DirectionRelation"/>
</ObjectPropertyDomain>
<ObjectPropertyDomain>
    <ObjectProperty URI="&qsr;object"/>
    <Class URI="&qsr;DistanceRelation"/>
</ObjectPropertyDomain>
<ObjectPropertyRange>
    <ObjectProperty URI="&qsr;object"/>
    <Class URI="&qsr;SpatialObject"/>
</ObjectPropertyRange>
<Declaration>
    <ObjectProperty URI="&qsr;object"/>
</Declaration>
<SubObjectPropertyOf>
    <ObjectProperty URI="&qsr;overlap"/>
    <ObjectProperty URI="&qsr;spatiallyRelated"/>
</SubObjectPropertyOf>
<InverseObjectProperties>
    <ObjectProperty URI="&qsr;overlap"/>
    <ObjectProperty URI="&qsr;overlap"/>
</InverseObjectProperties>
<Declaration>
    <ObjectProperty URI="&qsr;overlap"/>
</Declaration>
<ObjectPropertyDomain>
    <ObjectProperty URI="&qsr;spatiallyRelated"/>
    <Class URI="&qsr;SpatialObject"/>
</ObjectPropertyDomain>
```

```
<ObjectPropertyRange>
    <ObjectProperty URI="&qsr;spatiallyRelated"/>
    <Class URI="&qsr;SpatialObject"/>
</ObjectPropertyRange>
<Declaration>
    <ObjectProperty URI="&qsr;spatiallyRelated"/>
</Declaration>
<ObjectPropertyDomain>
    <ObjectProperty URI="&qsr;subject"/>
    <Class URI="&qsr;DirectionRelation"/>
</ObjectPropertyDomain>
<ObjectPropertyDomain>
    <ObjectProperty URI="&qsr;subject"/>
    <Class URI="&qsr;DistanceRelation"/>
</ObjectPropertyDomain>
<ObjectPropertyRange>
    <ObjectProperty URI="&qsr;subject"/>
    <Class URI="&qsr;SpatialObject"/>
</ObjectPropertyRange>
<Declaration>
    <ObjectProperty URI="&qsr;subject"/>
</Declaration>
<SubObjectPropertyOf>
    <ObjectProperty URI="&qsr;tangent_contains"/>
    <ObjectProperty URI="&qsr;spatiallyRelated"/>
</SubObjectPropertyOf>
<Declaration>
    <ObjectProperty URI="&qsr;tangent_contains"/>
</Declaration>
<SubObjectPropertyOf>
    <ObjectProperty URI="&qsr;tangent_inside"/>
    <ObjectProperty URI="&qsr;spatiallyRelated"/>
</SubObjectPropertyOf>
<InverseObjectProperties>
    <ObjectProperty URI="&qsr;tangent_inside"/>
    <ObjectProperty URI="&qsr;tangent_contains"/>
</InverseObjectProperties>
<Declaration>
    <ObjectProperty URI="&qsr;tangent_inside"/>
</Declaration>
<SubObjectPropertyOf>
    <ObjectProperty URI="&qsr;touches"/>
    <ObjectProperty URI="&qsr;spatiallyRelated"/>
</SubObjectPropertyOf>
<InverseObjectProperties>
    <ObjectProperty URI="&qsr;touches"/>
    <ObjectProperty URI="&qsr;touches"/>
</InverseObjectProperties>
<Declaration>
    <ObjectProperty URI="&qsr;touches"/>
</Declaration>
<ObjectPropertyDomain>
    <ObjectProperty URI="&qsr2;unit"/>
    <Class URI="&qsr;Scale"/>
</ObjectPropertyDomain>
```

```
<EntityAnnotation>
    <ObjectProperty URI="&qsr2;unit"/>
    <Annotation annotationURI="&rdfs;label">
        <Constant>unit</Constant>
    </Annotation>
</EntityAnnotation>
<Declaration>
    <ObjectProperty URI="&qsr2;unit"/>
</Declaration>
<DataPropertyDomain>
    <DataProperty URI="&qsr;maxDist"/>
    <Class URI="&qsr;Scale"/>
</DataPropertyDomain>
<DataPropertyRange>
    <DataProperty URI="&qsr;maxDist"/>
    <Datatype URI="&xsd;float"/>
</DataPropertyRange>
<Declaration>
    <DataProperty URI="&qsr;maxDist"/>
</Declaration>
<DataPropertyDomain>
    <DataProperty URI="&qsr;operator"/>
    <Class URI="&qsr;Metric"/>
</DataPropertyDomain>
<DataPropertyRange>
    <DataProperty URI="&qsr;operator"/>
    <Datatype URI="&xsd;string"/>
</DataPropertyRange>
<Declaration>
    <DataProperty URI="&qsr;operator"/>
</Declaration>
<ClassAssertion>
    <Class URI="&qsr;SmallScale"/>
    <Individual URI="&qsr;BuildingScale"/>
</ClassAssertion>
<Declaration>
    <Individual URI="&qsr;BuildingScale"/>
</Declaration>
<ClassAssertion>
    <Class URI="&qsr;LargeScale"/>
    <Individual URI="&qsr;CityScale"/>
</ClassAssertion>
<Declaration>
    <Individual URI="&qsr;CityScale"/>
</Declaration>
<Declaration>
    <Individual URI="&qsr;EuclidianDistance"/>
</Declaration>
<ClassAssertion>
    <Class URI="&qsr;MediumScale"/>
    <Individual URI="&qsr;NeiborhoodScale"/>
</ClassAssertion>
<Declaration>
    <Individual URI="&qsr;NeiborhoodScale"/>
</Declaration>
```

```
    <EntityAnnotation>
        <Individual URI="&qsr2;Kilometer"/>
        <Annotation annotationURI="&rdfs;label">
            <Constant>Kilometer</Constant>
        </Annotation>
    </EntityAnnotation>
    <Declaration>
        <Individual URI="&qsr2;Kilometer"/>
    </Declaration>
    <EntityAnnotation>
        <Individual URI="&qsr2;Meter"/>
        <Annotation annotationURI="&rdfs;label">
            <Constant>Meter</Constant>
        </Annotation>
    </EntityAnnotation>
    <Declaration>
        <Individual URI="&qsr2;Meter"/>
    </Declaration>
    <EntityAnnotation>
        <Individual URI="&qsr2;Mile"/>
        <Annotation annotationURI="&rdfs;label">
            <Constant>Mile</Constant>
        </Annotation>
    </EntityAnnotation>
    <Declaration>
        <Individual URI="&qsr2;Mile"/>
    </Declaration>
</Ontology>
```

DÉCOUVERTE D'ASSOCIATIONS SÉMANTIQUES POUR LE WEB SÉMANTIQUE GÉOSPATIAL : LE FRAMEWORK ONTOAST

RÉSUMÉ. Il est à présent communément admis que plus de 70% des pages Web contiennent des références spatiales et temporelles à travers l'occurrence de noms de lieux, d'adresses, de coordonnées géographiques, de dates, *etc.* Néanmoins, ces informations spatiales et temporelles restent inexploitées, alors qu'elles pourraient être utilisées par les moteurs de recherche pour préciser le contexte d'une requête, pour la désambiguïsation de celle-ci, pour la classification des résultats, *etc.*

Partant de ce constat, notre travail s'intéresse à l'étude des *techniques de représentation et de raisonnement* à base d'*annotations spatiales et temporelles*, indispensables pour la mise en place d'un futur Web Sémantique que l'on souhaiterait aussi Géospatial, c'est-à-dire capable par extension de gérer les dimensions spatiale mais également temporelle de l'information. L'objectif du Web Sémantique Géospatial est, en ce sens, identique à celui du Web Sémantique : associer aux données spatio-temporelles des descriptions (métadonnées) interprétables par les humains, mais surtout par les machines, afin que le traitement automatisé de ces données par des agents logiciels soit possible et efficient.

Nous définissons dans cette thèse un raisonneur spatial et temporel, compatible avec le langage standard de définition d'ontologies, OWL, ainsi qu'avec l'évolution OWL 2. Notre système, appelé ONTOAST, est capable d'exploiter à la fois des données spatiales et temporelles quantitatives (*i.e.* les coordonnées géométriques des objets spatiaux, les intervalles de temps ou/et les instants…) et des relations spatiales et temporelles qualitatives pour déduire des relations spatiales et temporelles jusque là implicites. Le but est de répondre à des questions telles que : "Quelles sont les villes qui se trouvent au *Sud-Ouest* de la France ?" ou "Quels sont les objectifs touristiques *proches* de ma position actuelle ?"…

Cette thèse s'intéresse également à une nouvelle technique de fouille du Web Sémantique, appelé analyse sémantique, qui vise la découverte des relations directes et indirectes qui existent entre deux individus. Nous proposons l'adaptation de l'analyse sémantique, initialement définie pour les graphes RDF(S), pour les ontologies OWL 2. Également, nous étendons ce processus d'analyse en y intégrant les informations spatiales et temporelles attachées aux individus. Celles-ci sont utilisées pour filtrer des connaissances ontologiques par rapport à un contexte spatial et temporel défini qui vise à préciser la portée spatiale et temporelle d'une requête, mais également pour la déduction de nouvelles associations sémantiques mettant en exergue une proximité spatiale ou/et temporelle entre individus.

MOTS CLÉS : *Web Sémantique Géospatial, ontologie, raisonnement spatial et temporel, analyse sémantique.*

ABSTRACT. It is now commonly accepted that over 70% of web pages contain spatial and temporal references through the use of place names, addresses, geographical coordinates, dates, *etc.* However, the temporal and spatial descriptions are currently unexploited by search engines, when they could be used in the search process for defining the context of a query, for query disambiguation, for result classification, *etc.* Based on this observation, our work focuses on the study of representation and reasoning techniques for spatial and temporal information in the context of the future Geospatial Semantic Web. The objective of the Geospatial Semantic Web is similar to that of the Semantic Web : attach to spatial and temporal data formal descriptions (metadata) that can be interpreted by humans, but mostly by machines, so that the automated processing of this data by software agents becomes possible and efficient.

We propose in this thesis a spatial and temporal reasoner, which is compatible with the standard ontology language OWL and with the evolution OWL 2. The system, called ONTOAST, is able to exploit both spatial and temporal quantitative data and spatial and temporal qualitative relations in order to infer implicit spatial and temporal qualitative relations. The goal is to answer questions such as : "What cities are located in the *southwest* of France ?", "What are the tourist attractions *near* my current position ?" …

This thesis also studies an alternative search paradigm, called *semantic analysis*, which aims the discovery of direct and indirect relationships existing between two individuals described using RDF(S) graphs. In order to infer additional semantic associations and to increase the accuracy of the analysis, we propose an adaptation of the *semantic analysis* for OWL 2 ontologies. We also show that new and possibly interesting semantic associations can be discovered, by taking into account spatio-temporal information which is usually attached to resources. Moreover, we propose to handle spatial and temporal contexts in order to limit the scope of the analysis to a region of space and a period of time. The *semantic analysis* discovery process uses ONTOAST for reasoning with spatial and temporal information and relations.

KEYWORDS : *Geospatial Semantic Web, ontology, spatial and temporal reasoning, semantic analysis.*

Alina Dia MIRON