Représentation et interrogation de connaissances dans des cartes cognitives spatio-temporelles

Adrian Robert

THÈSE DE DOCTORAT

UNIVERSITÉ D'ANGERS
COMUE UNIVERSITÉ BRETAGNE LOIRE

École Doctorale N° 601
Mathématique et Sciences et Technologies de l'Information et de la Communication
Spécialité : *Informatique*

Par

Adrian ROBERT

Représentation et interrogation de connaissances dans des cartes cognitives spatio-temporelles

Thèse présentée et soutenue à Angers, le 25 novembre 2020
Unité de recherche : LERIA (EA 2645)

Rapporteurs avant soutenance :

Fatma BOUALI
Patrice BUCHE

Composition du jury :

Président :	
Rapporteurs :	Fatma BOUALI, Professeur des Universités, Université de Lille
	Patrice BUCHE, Ingénieur de recherche - HDR, INRAE, IATE
Directeur de thèse :	Stéphane LOISEAU, Professeur des Universités, Université d'Angers
Co-encadrants de thèse :	David GENEST, Maitre de Conférences, Université d'Angers
	Brice TROUILLET, Professeur des Universités, Université de Nantes - LETG
	Thomas RAIMBAULT, Enseignant-Chercheur, EIGSI La Rochelle
Membres invités :	Fabrice GUILLET, Professeur des Universités, Université de Nantes
	Béatrice DUVAL, Professeur des Universités, Université d'Angers

Version pour les rapporteurs, compilée le 4 septembre 2020

TABLE DES MATIÈRES

INTRODUCTION GÉNÉRALE

L'intelligence artificielle (IA) vise à doter les machines d'intelligence. Le terme « intelligence artificielle » est cependant peu précis pour deux raisons principales. Premièrement il contient le mot « intelligence » qui est lui-même peu précis et embrasse un large panel de capacités cognitives : la modélisation, le raisonnement, la résolution, l'explication et caetera... Deuxièmement, le terme d'IA se trouve connoté par différents usages provenant de divers contextes, que ce soit des sciences, de l'art, de la technologie ou de l'économie. Le fait que tous ces usages changent au fil du temps rend ce terme d'« intelligence artificielle » polysémique.

Depuis la conférence de Dartmounth en 1956 [Moo06] que certains voient comme la naissance de la discipline, on distingue deux branches de l'IA. La première, nommée *IA numérique* utilise largement les statistiques et les probabilités tandis que la seconde, nommée *IA symbolique*, porte sur la représentation et le raisonnement principalement basés sur la logique. Ces deux branches ont eu à tour de rôle leurs périodes de prédilection dans l'histoire de l'IA.

Contexte

Notre travail de thèse se situe à l'intersection de la branche symbolique de l'intelligence artificielle et du domaine de l'interaction humain-machine. Notre thèse se situe dans la mouvance des travaux explicatifs de l'IA. Ces travaux ont pour but de fournir à l'utilisateur des connaissances appropriées pour comprendre les connaissances représentées ainsi que les raisonnements effectués par la machine. Notre thèse se situe aussi dans la mouvance des travaux d'interface humain-machine qui utilisent des modèles de cartes visuelles. Le modèle de cartes visuelles utilisé dans cette thèse est aussi un modèle de représentation de connaissances explicatif, c'est un modèle basé sur les graphes. Cette thèse se base ainsi sur le modèle visuel des *cartes cognitives* [Axe15 ; Tol48] qui est issu originairement de travaux en psychologie cognitive et est souvent utilisé en intelligence artificielle. Une carte cognitive est un graphe dont les noeuds sont étiquetés par des *concepts*, décrits par un texte bref, et dont les arcs sont des *influences* entre ces concepts, les arcs sont étiquetés par une valeur d'influence qui sert à qualifier l'influence. Les cartes cognitives sont utilisées dans des domaines de recherche ou des industries variés pour étudier un système d'influence complexe et éventuellement prendre des décisions s'inscrivant dans ce système [Ede92]. On les retrouve en sciences politiques [Axe15], en sciences sociales [Mag+13 ; Poi06], en géographie [Chr11], en écologie [ÖÖ03 ; ÖÖ04] ou en gestion [All96 ; Noh+00]. De manière générale les cartes cognitives sont conçues lors d'entretiens avec une personne dite « experte », les connaissances représentées par la carte sont les croyances de la personne experte. Une *taxonomie* de concepts, qui organise les concepts

grâce à une relation de spécialisation, peut être utilisée pour faciliter la construction des cartes en proposant un « vocabulaire », notamment pour la construction d'un corpus de cartes avec plusieurs experts. Le modèle des *cartes cognitives taxonomiques* [Cha10 ; CGL09 ; LeD13], sur lequel cette thèse se base, est une extension du modèle des cartes cognitives classiques auquel est associé une taxonomie. Le modèle des cartes cognitives taxonomiques permet d'effectuer des inférences simples comme l'*influence propagée* qui calcule l'influence de n'importe quel concept sur un autre de sorte à aider un *utilisateur* à analyser les cartes ou à prendre une décision.

Critiques et propositions

Cette thèse a démarré à l'issue du projet Kifanlo. Ce projet mené par des chercheurs en géographie vise à étudier les évolutions des stratégies de pêche en Loire-Atlantique depuis 50 ans. Pour cela, des pêcheurs ont été interrogés sur leur stratégies de pêche et une carte cognitive taxonomique par pêcheur interrogé a été construite. 53 cartes cognitives taxonomiques ont été récoltées, elles comprennent chacune entre 10 et 29 concepts. Deux critiques ont été adressées par les géographes concernant l'utilisation des cartes cognitives taxonomiques. Premièrement, la prise en compte de connaissances spatiales et temporelles n'est pas possible dans le modèle des cartes cognitives taxonomiques. Deuxièmement, l'exploitation d'un ensemble de cartes cognitives est difficile à maîtriser pour l'utilisateur. Notre thèse a pour objectif d'étendre le modèle des cartes cognitives taxonomiques pour répondre à ces deux critiques.

Concernant la première critique, le modèle des cartes cognitives ne permet en effet pas de représenter de connaissances temporelles ou spatiales autrement qu'à travers les noms des concepts. Cette lacune force l'utilisateur à modéliser les connaissances temporelles et spatiales de la même manière que les autres connaissances. Les connaissances spatiales et temporelles sont décrites de manière cachée dans les noms des concepts, ce qui fait que l'on ne les modélise pas réellement et on ne peut donc rien en inférer.

Pour répondre à cette limite, cette thèse propose un modèle de **cartes cognitives ontologiques**[1] qui utilise une ontologie regroupant des entités temporelles et spatiales pour caractériser les concepts des cartes. Cela permet accessoirement de réduire la complexité de la taxonomie de concepts, mais principalement de rendre ces connaissances temporelles et spatiales visualisables par l'humain et interprétables par la machine à des fins d'inférence. Aucun travail n'avait jusqu'à cette thèse proposé de modéliser le temps ou l'espace dans les concepts des cartes cognitives.

1. Le modèle de carte cognitive ontologique défini par Lionel Chauvin [Cha10 ; CGL09] a été renommé modèle de cartes cognitives taxonomiques dans les travaux utlérieurs. Dans notre thèse nous appelons également carte cognitive taxonomique le modèle de Chauvin et réservons le nom de cartes cognitives ontologiques à notre travail.

Concernant la seconde critique, l'exploitation par un utilisateur d'un ensemble de cartes de tailles conséquentes demeure effectivement fastidieuse. Bien que les graphes soient des structures appropriées à la visualisation, la quantité de connaissances que contient un ensemble de cartes rend difficile son exploitation visuelle.

Pour répondre à cette limite, cette thèse propose un **langage d'interrogation de cartes cognitives ontologiques** nommé CMQL pour Cognitive Map Query Language. Il est utilisé comme un outil d'interrogation des connaissances du modèle dans le but de permettre à un utilisateur d'accéder avec facilité, précision et exhaustivité aux diverses connaissances du modèle. Le langage CMQL permet en effet d'interroger n'importe quelle connaissance du modèle des cartes cognitives ontologiques, qui comprend les connaissances classiques des cartes cognitives, mais également les connaissances taxonomiques, spatiales et temporelles du modèle. CMQL a une syntaxe proche de SQL [Cha+76] et SPARQL [Pru08] et une sémantique basée sur le DRC (calcul relationnel de domaine) [LP82]. Le langage proposé se base sur un ensemble de primitives qui sont des relations entre les différents objets du modèle. Aucun travail n'avait jusqu'à cette thèse proposé d'interroger des cartes cognitives sous forme de requêtes.

Contributions

Les deux propositions principales faites dans cette thèse portent donc sur le modèle des cartes cognitives ontologiques et le langage d'interrogation pour ce modèle.

Le modèle des cartes cognitives ontologiques prend en compte les connaissances temporelles et spatiales selon un aspect de représentation de ces connaissances et un aspect de raisonnement sur ces connaissances.

Le premier aspect porte donc sur la représentation des connaissances temporelles et spatiales : les cartes cognitives ontologiques sont définies sur une *ontologie spatio-temporelle*. L'ontologie spatio-temporelle que l'on propose regroupe des connaissances sur le temps en étendant l'ontologie temporelle owl-time [PH04] et des connaissances sur l'espace en reprenant les ontologies spatiales de référence [BK11 ; PLA13 ; Sal+11]. Les *entités temporelles* utilisées pour étendre owl-time sont les *intervalles périodiques* [Osm99], ce sont des intervalles de temps qui se répètent périodiquement, comme les saisons ou les jours de travail. Ils ont étés choisis car ils sont adaptés au modèle des cartes cognitives qui ne représente généralement pas des concepts à des instants précis mais plutôt des événements récurrents ; c'est d'ailleurs le cas dans le projet Kifanlo. Les *entités spatiales* utilisées sont classiques [Gal09]. Des relations qualitatives, appelées *prédicats de comparaison*, sont utilisées pour définir et comparer les entités temporelles [Osm99] ou spatiales [Fra92 ; RC89] dans des *assertions spatio-temporelles* qui sont des triplets entité-prédicat-entité ; « l'Eté rencontre l'Automne » et « la

Bretagne touche la Normandie » sont des exemples d'assertions. Les connaissances contenues dans l'ontologie sont représentées par ces assertions. Les connaissances de l'ontologie spatio-temporelle peuvent alors servir à caractériser les concepts contenus dans les cartes cognitives. Pour cela, une carte cognitive contient aussi des assertions spatio-temporelles qui permettent de faire le lien entre les noeuds des cartes qui sont étiquetés par des concepts et l'ontologie. Ces assertions caractérisent temporellement et spatialement les noeuds des cartes et sont visualisables aisément sous les noeuds. Un noeud étiqueté par « Saison creuse » peut par exemple être caractérisé par l'assertion « la SaisonCreuse rencontre l'Ete ».

Le second aspect de la prise en compte du temps et de l'espace porte sur le raisonnement. En effet, l'intérêt principal de pouvoir représenter les connaissances spatio-temporelles de manière spécifique est de donner la possibilité à la machine de les manipuler. Les connaissances spécifiées par les utilisateurs sont spécifiques mais il est important de pouvoir accéder aux connaissances implicites. Par exemple si l'on sait que la période de pêche est d'Octobre à Juin, il est implicite que le mois d'Août ne fait pas partie de la période de pêche. Une inférence est nécessaire pour accéder à ces connaissances implicites. Cette inférence se base sur OWL [Hit+09] ainsi que sur un ensemble de règles d'inférence SWRL [Hor+04] provenant majoritairement de la littérature. Ces règles sont des règles de compositions sur les relations topologiques que ce soit pour le temps [BO00] ou pour l'espace [LY03], et des règles plus triviales comme celle de la transitivité sur les relations de distance.

Le langage d'interrogation de cartes cognitives ontologiques comporte deux aspects : le premier concerne un ensemble de *primitives* d'accès qui permettent de mettre en relation les différents objets du modèle, le deuxième aspect concerne le langage lui même qui utilise ces primitives d'accès.

Un ensemble de primitives d'accès est proposé pour accéder directement aux connaissances du modèle des cartes cognitives ontologiques. Une primitive est une relation entre les différents objets du modèle, comme les concepts, les chemins d'influence, les entités temporelles et caetera... Une *formule primitive* est une expression syntaxique désignant une primitive, elle peut être vue comme une requête très simple, par exemple fournir tous les concepts influencés par le concept « Navire moderne ». Une telle formule contient une constante ou une variable pour chaque attribut de la primitive, ce qui permet à l'utilisateur de contraindre le sens d'une primitive pour désigner des connaissances particulières.

Le langage d'interrogation, nommé *CMQL*, permet d'interroger de manière déclarative les connaissances du modèle des cartes cognitives ontologiques en utilisant les primitives d'accès et en les combinant entre elles grâce aux opérateurs classiques de la logique du premier ordre. CMQL possède une syntaxe issue de celle des langages SQL et SPARQL avec une forme générale du type : « SELECT variables FROM cartes WHERE formule », et une sémantique proche du calcul relationnel de domaine. CMQL permet d'accéder aux connaissances du modèle classique des cartes cognitives, ainsi qu'aux connaissances taxonomiques et aux

connaissances spatiales et temporelles.

Annonce du plan

Cette thèse est composée de 5 chapitres.

Le chapitre 1 est un chapitre bibliographique. Ce chapitre est composé de trois parties. La première partie compare cinq modèles de représentation de connaissances utilisant des graphes. La deuxième partie détaille le modèle de cartes cognitives taxonomiques sur lequel se base cette thèse. La troisième partie décrit l'utilisation des cartes cognitives dans le cadre du projet en halieutique Kifanlo.

Le chapitre 2 présente le modèle qui est proposé dans cette thèse, dénommé carte cognitive ontologique. Ce chapitre est composé de trois parties. La première partie décrit la représentation des connaissances temporelles. La deuxième partie décrit la représentation des connaissances spatiales. La troisième partie définit le modèle de cartes cognitives ontologiques.

Le chapitre 3 présente les primitives d'accès aux connaissances du modèle des cartes cognitives ontologiques. Ce chapitre est composé de quatre parties. La première partie concerne les primitives d'accès aux connaissances du modèle classique des cartes cognitives. La deuxième partie concerne les primitives d'accès aux connaissances taxonomiques. La troisième partie concerne les primitives d'accès aux connaissances temporelles. La quatrième partie concerne les primitives d'accès aux connaissances spatiales.

Le chapitre 4 présente le langage de requêtes CMQL. Ce chapitre est composé de quatre parties. La première partie est une partie bibliographique qui présente deux langages d'interrogation de modèles graphiques : SPARQL et Cypher. La deuxième partie présente la syntaxe de CMQL. La troisième partie fournit un exemple de requête permettant d'expliquer le fonctionnement des variables libres dans le langage. La quatrième partie présente la sémantique de CMQL de manière formelle puis au travers d'un exemple.

Le chapitre 5 concerne le côté applicatif de cette thèse, il est composé de deux parties. La première partie présente des données issues du projet Kifanlo et montre comment nos outils peuvent les exploiter. La deuxième partie présente l'outil VSPCC qui est le logiciel repris de [LeD13] et sur lequel nous avons intégré les cartes cognitives ontologiques et le langage de requête.

Le coeur de la thèse est constitué des chapitres 2, 3 et 4. La lecture du chapitre 2 nécessite la connaissance du modèle des cartes cognitives taxonomiques présentées dans les parties 1.1.3 et 1.2.

CHAPITRE 1

LE MODÈLE DES CARTES COGNITIVES

Introduction

Les représentations de connaissances utilisant des graphes ne sont pas récentes. Une des plus anciennes, souvent citée comme étant la base des taxonomies modernes [Sow87 ; Ver14], est l'arbre de Porphyre(figure 1.1).

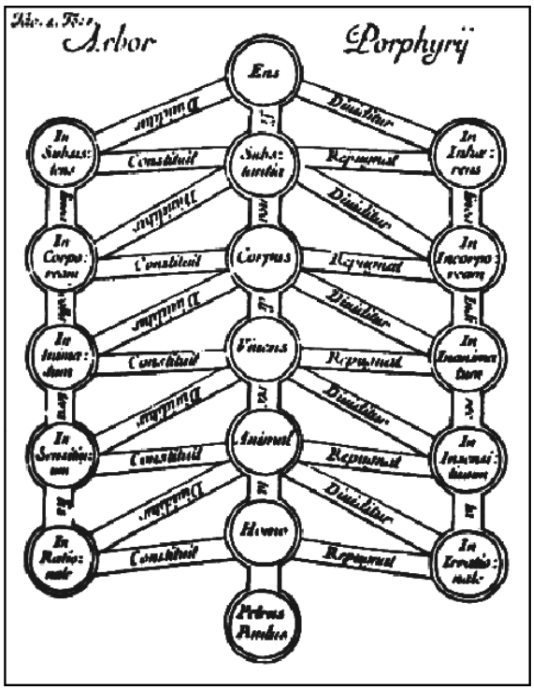


FIGURE 1.1 – Arbre de Porphyre

Tout comme l'arbre de Porphyre, de nombreux modèles sont basés sur les graphes pour représenter des connaissances. Parmi ces modèles, on distingue les modèles formels de *réseaux sémantiques* [Bra79 ; Leh92 ; Sow87 ; Woo75] provenant de l'intelligence artificielle et les modèles de *cartographie sémantique*, moins formels et plus visuels, provenant de divers domaines tels que l'éducation [GN84], la psychologie [Tol48] ou les sciences sociales [Axe15].

Les réseaux sémantiques ont émergé dans le domaine de la traduction automatique,

lorsque des chercheurs de Cambridge dans les années 60 ont utilisé un « *interlingua* »[Mas61], semblable à un arbre syntaxique abstrait, dont le but était de faire le lien entre deux langages naturels. De nombreux types de réseaux sémantiques ont vu le jour [Sow87]. On peut citer : 1) les réseaux définitionnels qui se concentrent historiquement sur les relations de généralisation/subsombtion pour représenter des ontologies, 2) les réseaux propositionnels qui sont des bases de faits ou de propositions comme par exemple le modèle RDF(Ressource Description Framework) [MMM+04], 3) les réseaux implicationnels qui utilisent l'implication, la cause ou l'influence comme relation principale, comme par exemple les réseaux bayésiens [Pea88], les réseaux causaux [Rie76] ou réseaux de croyances [Doy79], 4) les réseaux exécutables qui ont une certaine capacité calculatoire, et proposent des mécanismes d'inférence comme les réseaux de Petri [Pet66]. Un grand nombre de modèles font partie de réseaux hybrides, ils combinent plusieurs types de réseaux sémantiques, comme les réseaux bayésiens qui sont implicationnels et exécutables.

Les modèles de cartographie sémantique ont un but directement applicatif et sont en conséquence plus simples et visuels. C'est le cas des cartes cognitives [Axe15 ; Tol48] qui peuvent s'apparenter aussi aux réseaux implicationnels de l'IA. Le modèle des cartes cognitives est un modèle visuel simple de cartographie sémantique qui représente des stratégies ou plus généralement des systèmes d'influence. Les cartes cognitives ont des origines dans la biologie et les sciences sociales et sont couramment utilisées pour l'aide à la prise de décision ou l'étude des perceptions que certains acteurs peuvent avoir d'un système donné.

Ce chapitre est composé de trois parties. La première partie fournit une présentation de divers modèles de cartographie sémantique et de réseaux sémantiques ; le modèle des cartes cognitives sur lequel est basé la suite de la thèse y est défini dans la troisième sous-partie. La deuxième partie présente le modèle de cartes cognitives taxonomiques, qui constitue la base des recherches de cette thèse. La troisième partie traite de l'utilisation des cartes cognitives dans le domaine de la recherche en géographie marine dans le cadre du projet KIFANLO.

1.1 Présentation de modèles

Cette partie consiste en une présentation de différents modèles de représentation de connaissances à base de graphe. Cinq modèles sont ici présentés pour être ensuite comparés selon quatre critères :

— Le type de connaissances représentées.

— Les objectifs et applications du modèle.

— La construction et l'utilisation du modèle.

— L'importance de la sémantique et de l'aspect visuel du modèle.

Nous décrivons en premier lieu les logiques de descriptions issues des réseaux sémantiques et particulièrement en vogue dans la recherche en IA symbolique, nous présentons ensuite les

réseaux bayésiens faisant partie de la famille des réseaux implicationnels. Les trois modèles suivants proviennent de la cartographie sémantique : les cartes cognitives pouvant être une aide à la prise de décision et à l'analyse de systèmes complexes, les cartes conceptuelles axées sur l'aide à l'apprentissage et les cartes mentales servant à la prise de notes et à la mémorisation. Les modèles sont enfin confrontés selon les quatre critères de comparaison.

1.1.1 Logiques de description

Les logiques de description [Baa+03] sont une famille de langages de représentation de connaissances provenant des langages à cadres [Min74] et des réseaux sémantiques [Bra79 ; Leh92 ; Sow87 ; Woo75]. Le nom de logique de description vient d'une part de la description des concepts utilisée pour décrire un domaine de connaissance, d'autre part de la sémantique basée sur la logique des prédicats du premier ordre. Celle-ci est indécidable, c'est pour cette raison que les logiques de description sont une famille contenant divers langages réduisant l'expressivité de la logique des prédicats du premier ordre. Cela permet d'en isoler des fragments décidables ayant des complexités différentes pour les problèmes de subsombtion, de satisfiabilité et de vérification d'instance. Les logiques de descriptions sont proches du modèle des graphes conceptuels [MC92 ; Sow76], issus des réseaux sémantiques et se basant également sur la logique du premier ordre, certaines variantes sont plus expressives et donc indécidables. La structure de graphe a plus d'importance dans le modèle des graphes conceptuels que dans celui des logiques de description, la plupart des opérations de ce modèle sont des opérations de graphes, comme par exemple la projection, une opération importante des graphes conceptuels, qui est un morphisme de graphe.

Sémantique

La description des concepts s'effectue récursivement à partir de concepts atomiques, de rôles atomiques et des *constructeurs* dont dispose le langage. Le choix des constructeurs différencie les logiques de descriptions les unes des autres, ainsi le nom de la logique de description nous indique les constructeurs choisis et donc l'expressivité du langage. Le tableau ci-dessous indique le sens des différentes lettres composant les noms des logiques de description.

Lettre	*Constructeur*	*Syntaxe*	*Sémantique*
$\mathcal{AL}$	*Nom de concept*	C	$C^{\mathcal{I}}$
$\mathcal{AL}$	*Top*	$\top$	$\Delta^{\mathcal{I}}$
$\mathcal{AL}$	*Bottom*	$\bot$	$\varnothing$
$\mathcal{AL}$	*Nom de role*	R	$R^{\mathcal{I}}$
$\mathcal{AL}$	*Conjonction*	$C_1 \sqcap C_2$	$C_1^{\mathcal{I}} \cap C_2^{\mathcal{I}}$
$\mathcal{AL}$	*Quantifieur universel*	$\forall R.C$	$\{d_1 \in \Delta^{\mathcal{I}} \mid \forall d_2 \in \Delta^{\mathcal{I}}, (R^{\mathcal{I}}(d_1, d_2) \rightarrow d_2 \in C^{\mathcal{I}})\}$
$\mathcal{C}$	*Négation de concept*	$\neg C$	$\Delta^{\mathcal{I}} \backslash C^{\mathcal{I}}$
$\mathcal{U}$	*Disjonction*	$C_1 \sqcup C_2$	$C_1^{\mathcal{I}} \cup C_2^{\mathcal{I}}$
$\mathcal{E}$	*Quantifieur existentiel*	$\exists R.C$	$\{d_1 \in \Delta^{\mathcal{I}} \mid \forall d_2 \in \Delta^{\mathcal{I}}, (R^{\mathcal{I}}(d_1, d_2) \wedge d_2 \in C^{\mathcal{I}})\}$
$\mathcal{N}$	*Restriction de cardinalité*	$\leqslant n\ R$	$\{d_1 \in \Delta^{\mathcal{I}} \mid \{d_2 \mid R^{\mathcal{I}}(d_1, d_2)\} \leqslant n\}$
$\mathcal{Q}$	*Restriction qualifiée*	$\leqslant n\ R.C$	$\{d_1 \in \Delta^{\mathcal{I}} \mid \{d_2 \mid R^{\mathcal{I}}(d_1, d_2), d_2 \in C^{\mathcal{I}}\} \leqslant n\}$
$\mathcal{R}$	*Conjonction de rôles*	$R_1 \sqcap R_2$	$R_1^{\mathcal{I}} \cap R_2^{\mathcal{I}}$
$\mathcal{H}$	*Hiérarchie de rôles*	$R_1 \sqsubseteq R_2$	$R_1^{\mathcal{I}} \subseteq R_2^{\mathcal{I}}$
$\mathcal{I}$	*Inversion de rôles*	R^{-1}	$\{(d_1, d_2) \in \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \mid R^{\mathcal{I}}(d_2, d_1)\}$
$\mathcal{R}^+$	*Transitivité de rôles*	R^+	$\bigcup (R^{\mathcal{I}})^i$
$\mathcal{O}$	*un de*	$\{a_1, .., a_n\}$	$\{d \in \Delta^{\mathcal{I}} \mid d = a_i^{\mathcal{I}}\ pour\ un\ a_i\}$

Ainsi la logique de description $\mathcal{ALC}$ [Baa+03 ; SS91] utilise les 5 premiers constructeurs ($\mathcal{AL}$) et la négation de concept ($\mathcal{C}$). Certains ensembles de constructeurs sont équivalents, c'est le cas des logiques $\mathcal{ALC}$ et $\mathcal{ALUE}$. Certaines lettres absentes de ce tableau sont utilisées comme raccourcis. Par exemple la lettre $\mathcal{S}$ que l'on peut trouver notamment dans le nom de la logique $\mathcal{SHOIN}$ utilisée par OWL DL [Hit+09 ; HP03] est un raccourci pour les constructeurs de $\mathcal{ALC}$ et la transitivité des rôles ($\mathcal{R}^+$).

Les logiques de description définissent leur sémantique au travers d'une *interprétation* d'une *signature*. La signature comprend les concepts, les rôles et les individus. L'interprétation fait correspondre les concepts à un sous-ensemble d'un domaine donné, les rôles à des relations binaires sur un domaine donné et les individus à des éléments de ce domaine.

Définition 1 (Interprétation)**.** *Soit N_C={A,B,C...} un ensemble de concepts atomiques, N_R={r,s,t...} un ensemble de rôles atomiques et N_I={a,b,c...} un ensemble d'individus. Si N_C, N_R et N_I sont finis et distincts deux à deux alors $\mathcal{S} =< N_C, N_R, N_I >$ est une signature. Une interprétation $\mathcal{I}$ pour $\mathcal{S}$ est un couple $\mathcal{I} = (\Delta^{\mathcal{I}}, .^{\mathcal{I}})$ tel que :*

- *$\Delta^{\mathcal{I}}$ est un ensemble non vide, le domaine d'interprétation.*
- *$.^{\mathcal{I}}$ est une fonction affectant :*
  - *un sous-ensemble $C_i^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$ à chaque concept atomique $C_i \in N_C$,*
  - *une relation $R_i^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ à chaque rôle atomique $R_i \in N_R$,*

— *et un élément* $a_i^{\mathcal{I}} \in \Delta^{\mathcal{I}}$ *à chaque individu* $a_1 \in N_I$.

Traditionnellement, les logiques de description divisent la connaissance en deux parties : Les connaissances terminologiques (TBox) sont des connaissances générales, ontologiques, concernant des concepts et des relations entre eux.
Les connaissances sur les individus (ABox) sont des connaissances spécifiques à des individus, ce sont des faits.
Une base de connaissances est l'ensemble de ces deux parties : la TBox et la ABox.

Étant donné une logique de description $\mathcal{L}$ et une signature $\mathcal{S}$, une base de connaissances Σ dans $\mathcal{L}$ est une paire $\Sigma = \langle T, A \rangle$ telle que :

— T est la TBox, un ensemble fini, qui peut être vide, d'expressions de la forme $C_1 \sqsubseteq C_2$ telle que C_1 et C_2 sont des concepts ou de la forme $R_1 \sqsubseteq R_2$ telle que R_1 et R_2 sont des roles. $C_1 := C_2$ est une notation pour $C_1 \sqsubseteq C_2$ et $C_2 \sqsubseteq C_1$. Les formules de T sont appelées axiomes terminologiques.

— A est la ABox, un ensemble fini, qui peut être vide, d'expressions de la forme $a : C$ ou $(a, b) : R$, où C est un concept, R est un rôle qui n'est pas nécessairement atomique, et a, b appartiennent à R_I. Les formules de A sont appelées des assertions.

		$TBox$
Femme	$\sqsubseteq$	Personne
Homme	:=	Personne $\sqcap \neg$Femme
Mere	:=	Femme $\sqcap \exists$aUnEnfant.Personne
Pere	:=	Homme $\sqcap \exists$aUnEnfant.Personne
Parent	:=	Mere $\sqcup$ Pere
		$ABox$
		Femme(Marie)
		aUnEnfant(Marie,Jean)
		Homme(Jean)
		aUnEnfant(Pierre,Jean)

Exemple 1. *Le tableau ci-dessus représente un exemple de TBox et de Abox. Les connaissances de la TBox sont terminologiques, générales tandis que celles de la ABox sont assertionnelles, spécifiques à des individus. Les opérateurs utilisés sont l'inclusion, l'équivalence, la conjonction, la disjonction, la négation simple et la quantification existentielle. Ces expressions sont donc valides pour la plupart des logiques de descriptions.*

Inférence

Les logiques de descriptions permettent non seulement de représenter formellement des connaissances mais également d'en inférer de nouvelles n'étant pas présentes explicitement.

Grâce à des taches de raisonnement, l'inférence peut s'appliquer aux connaissances terminologiques comme aux connaissances assertionnelles. On pourra notamment vérifier qu'un concept est satisfaisable par rapport à une TBox ou s'il subsume un autre concept. On pourra également inférer toutes les appartenances d'une instance à des concepts.

Exemple 2. *En reprenant la TBox et la ABbox de l'exemple précédent, il est possible d'en inférer les connaissances suivantes : Mere(Marie), Parent(Marie), Pere(Jean), Parent(Jean)...*

Étudions comment se situe ce modèle par rapport aux critères de comparaison.

Types de connaissances Les logiques de descriptions sont particulièrement expressives, elles permettent de représenter des connaissances sur les domaines « naturels » avec précision. Elles sont cependant moins efficaces pour représenter d'autres domaines comme par exemple les mathématiques et les équations algébriques [Baa+03].

Objectifs et application Les logiques de description servent principalement à décrire avec précision un domaine particulier et effectuer des raisonnements sur ces connaissances. Elles ont beaucoup d'applications[Hor05a], comme en configuration [MW98], en ingénierie logicielle [Dev+91] ou en traitement automatique des langues [Bus+06 ; Fra94], mais elles sont particulièrement connues pour être la base de langages d'ontologies comme OIL ou OWL [HPV03]. Or ces ontologies jouent un rôle majeur dans le web sémantique[BHL01], et sont très utilisées dans des domaines tels que l'e-Science [GOR09 ; HDN04], la bio-informatique [SGB00 ; Wro+03] et le médical [Ped+12 ; Rec03].

Construction et utilisation L'implémentation d'un système de représentation de connaissance basé sur les logiques de description est compliquée, ce qui n'est pas étonnant étant donnée la complexité de ses composants (interface, raisonneurs fiables, optimisations...). De nombreux systèmes de ce genre ont été développés et sont disponibles au public pour une utilisation relativement aisée [Baa+03]. Il existe par ailleurs des ressources décrivant des méthodologies et bonnes pratiques pour la construction d'ontologies [Baa+03].

Aspect visuel et sémantique L'aspect visuel n'est donc pas un point fort de ce modèle. Bien que les logiques de descriptions puissent être représentées visuellement sous forme de graphe [HS03], ce n'est pas l'objectif du modèle, l'accent étant mis sur la formalisation claire de la sémantique et les taches de raisonnement.

1.1.2 Réseaux bayésiens

Les réseaux bayésiens [Pea88] sont des réseaux implicationnels, c'est-à-dire qu'ils modélisent une sorte d'implication ou influence entre différentes variables. En l'occurrence, les réseaux bayésiens modélisent des dépendances probabilistes entre variables sous forme de graphe. C'est donc à la fois un modèle graphique et un modèle probabiliste. Les réseaux bayésiens peuvent également être considérés comme des réseaux exécutables puisque un intérêt majeur de ce modèle est le calcul d'inférence probabiliste permettant d'obtenir les probabilités de variables connaissant l'état d'autres variables.

Sémantique

Les variables représentées par des noeuds ont une distribution de probabilité conditionnée par l'état d'autres variables représentées par les noeuds parents dans le graphe. Pour cela, à chaque variable est associée une table de probabilité sachant l'état de ses parents. Les arcs représentent une relation de dépendance probabiliste entre deux variables, ce qui rend le modèle visuellement intéressant car même si on omet les tables de probabilités conditionnelles, la structure du graphe représente clairement les dépendances entre variables.

Définition 2. *Un réseau bayésien $\mathcal{B}$ est un couple ($\mathcal{G} = (X, E), \mathcal{P}$) où :*

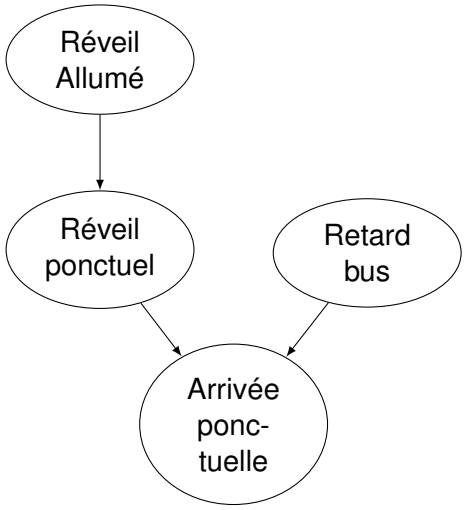
- *$\mathcal{G}$=(X,E) est un graphe orienté acyclique. X et E sont respectivement les ensembles des sommets et des arcs de $\mathcal{G}$.*
- *$\mathcal{P}$ une collection de probabilités conditionnelles telle qu'à chaque variable aléatoire représentée par un sommet X_i est associée une table de probabilités conditionnelles $\mathbb{P}(X_i|Pa(X_i))$ où Pa(X_i) est l'ensemble des parents de X_i dans $\mathcal{G}$.*
- *La distribution de probabilité jointe est vérifiée : $\mathbb{P}(X_1, ..., X_n) = \prod_{i=1}^{n} \mathbb{P}(X_i|Pa(X_i))$.*

Exemple 3. *Cet exemple décrit le réseau bayésien de la figure 1.2. C'est un graphe constitué de quatre noeuds représentant des variables ayant deux états possibles, vrai ou faux, comme la variable* retard d'un bus*, le bus pouvant soit être à l'heure soit en retard. Les arcs entre ces noeuds représentent les dépendances de probabilités conditionnelles entre ces variables, comme l'arc allant de* retard de bus *à* arrivée ponctuelle *qui signifie que la probabilité d'*arrivée ponctuelle *dépend du* retard du bus*. À chaque sommet est associée une table de probabilité conditionnelle.* Réveil allumé *et* Retard bus *sont représentés par des sommets n'ayant pas de parents, ces variables ne dépendent donc d'aucune autre variable, leur table de probabilité contient alors des probabilités a priori. Il a été défini pour l'exemple que la probabilité que le réveil soit allumé soit de 0.9 et la probabilité que le bus soit en retard soit de 0.1. Les deux autres variables,* Reveil ponctuel *et* Arrivée ponctuelle*, dépendent, elles, d'autres variables.* Réveil ponctuel *dépend de* Réveil allumé *et sa table de probabilité est conditionnée par l'état de* Réveil allumé*, ainsi la probabilité que le réveil soit ponctuel est de 0.9 quand le réveil a été allumé et de 0.3 quand il n'a pas été allumé. La variable* Arrivée ponctuelle *dépend de deux variables,* Reveil ponctuel *et* Retard bus*. Sa table de probabilité est alors composée des probabilités conditionnées par les états de ces deux variables. Par exemple si le réveil est ponctuel et que le bus est en retard la probabilité d'arriver à l'heure est de 0.6.*

Il existe plusieurs variantes de réseaux bayésiens, certains ne modélisent pas des variables discrètes mais continues, d'autres modélisent les deux [Lau92 ; Mur98]. Pour des variables continues, des fonctions de densité de probabilité sont alors utilisées à la place des tables de probabilités conditionnelles. En reprenant l'exemple du retard de bus, on peut s'imaginer que la variable soit continue et que son état soit la durée du retard en minutes. Quand des variables

Réveil allumé	
T	F
0.9	0.1

R. allumé	Réveil ponctuel	
	T	F
F	0.3	0.7
T	0.9	0.1



Retard bus	
T	F
0.1	0.9

Retard bus	Réveil ponctuel	Arrivée ponctuelle	
		T	F
F	F	0.4	0.6
F	T	1	0
T	F	0.1	0.9
T	T	0.6	0.4

FIGURE 1.2 – Exemple de réseau bayésien

discrètes et continues sont utilisées, des contraintes supplémentaires sont à respecter, par exemple le fait que les variables discrètes ne puissent pas dépendre de variables continues.

Inférence

La sémantique forte des réseaux bayésiens, basée sur les probabilités conditionnelles, offre un certain nombre de propriétés permettant d'inférer des connaissances. Parmi ces propriétés on citera l'indépendance conditionnelle qui nous permet d'affirmer par exemple que sachant *Réveil ponctuel*, alors *Réveil allumé* et *Arrivée ponctuelle* sont des variables indépendantes, ou encore que ne sachant pas *Arrivée ponctuelle* alors les variables *Réveil ponctuel* et *Retard bus* sont indépendantes. La propriété de d-séparation généralise l'indépendance conditionnelle en prenant en compte un ensemble de variables connues. L'inférence dans un réseau bayésien permet principalement soit de déterminer les indépendances entre variables, soit de déterminer les probabilités a posteriori des variables connaissant l'état d'autres variables, souvent grâce à des observations. L'inférence des probabilités a posteriori permet de répondre à n'importe quelle question concernant la valeur de variables du réseau bayésien selon des observations. L'on peut par exemple se demander quelle serait la probabilité d'arriver à l'heure sachant que le réveil est allumé et que le bus est en retard, ce qui nous donnerait une probabilité inférée de 0,55 [1]. L'on pourrait également se demander l'inverse : Quelle serait la probabilité que le réveil soit allumé et que le bus soit en retard si l'on arrive à l'heure ? Cela nous donnerait une probabilité inférée très faible de 0,056 [2]. Ce problème d'inférence est cependant NP-difficile, le rendant difficilement utilisable sur des gros graphes. Il existe malgré tout plusieurs algorithmes efficaces retournant des valeurs approchées comme l'algorithme d'acceptance-rejet ou l'algorithme Metropolis-Hastings. Les réseaux bayésiens sont également utilisés en apprentissage automatique, notamment grâce à des méthodes d'apprentissage des paramètres (c'est-à-dire des tables de probabilité) à partir de jeux de données [FG99], ou alors d'apprentissage de la structure du graphe [SV95] ce qui permet d'obtenir des dépendances entre variables.

Étudions comment se situe ce modèle par rapport aux critères de comparaison.

Types de connaissances Les réseaux bayésiens sont focalisés sur la représentation de probabilités conditionnelles et dépendances probabilistes entre variables aléatoires. Les connaissances représentées sont de nature spécifique, ce modèle n'a pas vocation à décrire de manière générale des domaines variés comme le font les logiques de description.

Objectifs et applications Ce modèle permet de représenter des dépendances probabilistes et d'inférer des probabilités conditionnelles pour effectuer des prédictions. C'est pourquoi ils sont utilisés dans de nombreux domaines tels que le diagnostic industriel ou médical [Kah+97], le traitement d'images [DLN06], la détection de fraude, de spams [And+00] ou même la météo-

1. $\mathbb{P}(AP|ra, rb) = \frac{\mathbb{P}(AP,ra,rb)}{\mathbb{P}(ra,rb)} = \frac{0,0495}{0,09} = 0,55$
2. $\mathbb{P}(RA, RB|ap) = \alpha \sum_{RP} \mathbb{P}(ra, rb, rp, ap) = 1,134 \times 0,00495 = 0,056$

rologie [CSG04].

Construction et application Un réseau bayésien est constitué de deux éléments : sa structure, représentée par le graphe, et ses paramètres qui sont les tables de probabilités conditionnelles. La construction de ce modèle est donc effectuée en deux étapes. Il s'agit d'abord de déterminer la structure, ensuite les paramètres. Ces connaissances proviennent généralement des experts du domaine concerné mais il est possible de les obtenir à partir de données, notamment pour les paramètres qui sont souvent calculés empiriquement.

Aspect visuel et sémantique Le côté visuel des réseaux baysiens est assez important, car il permet efficacement de connaître les dépendances entre variables aléatoires. La taille du réseau bayésien, pouvant en pratique être considérable, peut néanmoins être un frein. La sémantique de ce modèle est très précise, les sommets et les arcs ayant une signification mathématique, ou plutôt probabiliste, cela permet d'effectuer des raisonnements sur ce modèle.

1.1.3 Cartes cognitives

Les cartes cognitives [Axe15 ; Tol48] représentent des influences entre différents concepts. Elles peuvent donc être rapprochées des réseaux d'influence et des réseaux bayésiens mais font partie de la cartographie sémantique. Elles sont utilisées pour l'aide à la prise de décision ou l'étude des perceptions de systèmes sociaux complexes [CA92 ; Ede92 ; ÖÖ04].

Le terme de « carte cognitive »est apparu en psychologie cognitive, lorsque Edward C Tolman l'a employé en 1948 dans le cadre de recherche sur les mécanismes d'apprentissage. Tolman a ainsi placé des rats dans un labyrinthe pour les mettre en situation d'apprentissage. Le terme de carte cognitive se réfère ici à la représentation spatiale qu'ont les rats de leur environnement. Il montra que leur carte cognitive avait une incidence particulière sur leur comportement. Tolman n'a cependant fourni ni formalisation ni représentation concrète des cartes cognitives.

C'est Robert Axelrod, chercheur en sciences politiques, qui fournit en 1976 la première formalisation de cartes cognitives en les présentant comme un modèle de représentation graphique permettant d'exprimer les croyances d'un individu concernant un problème particulier. Les cartes cognitives d'Axelrod ne concernent plus seulement la représentation spatiale d'un individu mais un ensemble de croyances qu'a un individu du monde réel. Il propose également un mécanisme d'inférence permettant de calculer l'influence d'un concept sur un autre.

Les cartes cognitives ont une nouvelle fois évolué avec James Doyle et David Ford [DF99], n'ayant plus la prétention de représenter la pensée, mais d'être une représentation graphique de structure analogue à celle perçue d'un système complexe. Dans leur article, ils font état des différentes utilisations du terme carte cognitive, utilisé dans divers domaines avec des sens distincts. Les cartes cognitives peuvent d'ailleurs être appelées « cartes causales », « graphes causaux », « graphes cognitifs » ou « modèles mentaux ».

Dans le domaine de l'intelligence artificielle, Kosko définit aussi son modèle de cartes cognitives qu'il appelle cartes cognitives floues [Kos+86] en se basant sur la logique floue [Zad88]. L'incertitude dans les cartes cognitives, résidant principalement dans les influences, est désormais placée dans les concepts qui sont alors des concepts flous. Mis à part sa sémantique provenant de la logique floue, ce modèle diffère des cartes cognitives classiques de par sa capacité d'apprentissage automatique et ses méthodes de simulation/scénarios qui lui proviennent de nombreuses recherches en intelligence artificielle [PSG04 ; PSG06 ; SKP05]. Depuis les années 2000, la plupart des recherches en intelligence artificielle concernant les cartes cognitives se basent sur les cartes cognitives floues. Certaines extensions ont même été apportées aux cartes cognitives floues, telles que les réseaux cognitifs dynamiques [Mia+01], les cartes cognitives floues à base de règles [CT99], ou encore les cartes cognitives floues aléatoires [Agu02]. Des recherches en intelligence artificielle continuent de se baser sur le modèle classique de cartes cognitives et l'ont étendu, notamment en y rattachant une taxonomie et en proposant des méthodes de manipulation comme les vues ou les synthèses.

Une carte cognitive représente des influences entre concepts, c'est un graphe dont les noeuds sont étiquetés par des concepts et les arcs sont des influences. Un concept est un texte bref qui le caractérise, une influence est une relation causale d'un concept à un autre caractérisée par une valeur d'influence. Toutes les influences d'une carte cognitive ont une valeur d'influence qui appartient à un même ensemble de valeurs sur lequel est définie la carte cognitive. Cet ensemble de valeurs est à définir en amont de la construction des cartes, le choix de celui-ci peut être varié et dépend de l'utilisation, du niveau de détail désiré. Par exemple il est possible d'utiliser un ensemble de valeurs symboliques et simples comme l'ensemble $\{+, -\}$ défini par Axelrod ou l'ensemble $\{nul < faible < moyen < fort\}$. Cet ensemble de valeurs peut également être un intervalle numérique comme [-1 ;1].

Définition 3 (Carte cognitive)**.** *Soit C un ensemble de concepts, soit I un ensemble de valeurs. Une carte cognitive définie sur I est un graphe orienté étiqueté $CM = (N, A, labelN, labelE)$ où :*

- *N est l'ensemble des noeuds du graphe.*
- *$A \subseteq N \times N$ est l'ensemble des arcs appelés influences.*
- *$labelN : N \rightarrow C$ est une fonction d'étiquetage qui associe à chaque noeud un concept de C.*
- *$labelE : A \rightarrow I$ est une fonction d'étiquetage qui associe à chaque influence une valeur d'influence.*

Exemple 4. *La carte cognitive de la figure 1.3 est définie sur l'ensemble de valeurs proposé par Axelrod : {+,-}. Cette carte cognitive représente un système d'influence lié au filtrage d'internet. L'influence allant du concept* `Filtrage Internet` *vers le concept* `Information pro-régime` *est*

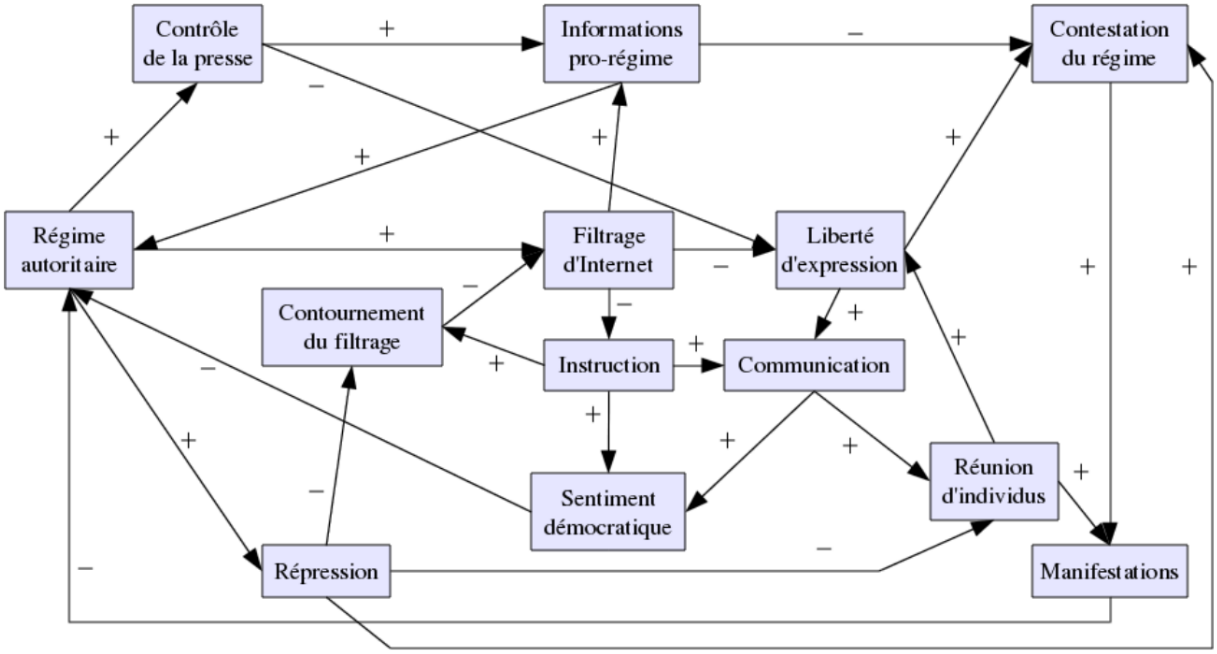


FIGURE 1.3 – Exemple de carte cognitive représentant un système d'influence lié au filtrage d'internet [LeD13]

étiquetée par la valeur « + » : cela signifie que le filtrage d'internet influence positivement les informations pro régime et en augmente la quantité. À l'inverse, l'influence allant du concept `Filtrage Internet` *vers le concept* `Liberté d'expression` *est étiquetée par la valeur « - » : cela signifie que le filtrage d'internet influence négativement la liberté d'expression et donc la limite.*

Sémantique des influences

Dans les cartes cognitives, les influences sont au coeur du modèle. Nous tentons ici d'apporter quelques explications quant à leur sémantique, en nous basant sur d'autres travaux mais aussi sur des avis plus personnels.

Une influence est parfois vue comme une cause, dans un sens assez large où la cause ne serait pas unique et serait donc différente d'une implication et ne garantirait pas sa contraposée [Kos+86]. Mais une influence n'est pas une cause, du moins pas seulement une cause [Cos08] : un modèle de carte cognitive ne représentant que des causes est d'ailleurs appelé carte causale [BWB77]. Une influence est un terme vague, qui englobe plus que l'idée de cause. De manière générale une influence d'un concept A vers un concept B signifie que le concept A agît sur le concept B, qu'il *peut entraîner une modification* de B. Certaines influences peuvent aussi être vues différemment dans le cas où elles représenteraient une relation de

« moyen » à « fin », c'est-à-dire une intention du sujet de la carte [All96] (par exemple : « les bénéfices influencent/permettent les investissements »).

De manière générale, une influence reste ambiguë sur trois points :

— Les objets de l'influence : quelles propriétés de A influencent quelles propriétés de B ? La publicité m'influence, mais qu'est-ce qu'elle influence ? Influence-t-elle mon poids, ma fatigue, mes réponses émotionnelles ? De la même manière, on peut se poser la question : « Quelle propriété de la publicité m'influence ? ». Serait-ce les offres d'emploi de son marché, ou les stimuli provenant de ses supports de communication ?

— La nature de l'influence, c'est-à-dire la nature de la modification entraînée. Une influence n'est pas une cause, elle peut agir ou entraîner une modification, un effet. On peut alors se demander comment, par quel procédé un effet est-il rendu ? Et quel est cet effet ? Prenons le cas de la publicité qui m'influence : l'influence consiste-t-elle en l'augmentation temporaire d'un état d'énervement suite à la coupure de mon programme radio préféré, ou plutôt en l'altération de mes processus de décision aux travers de stimuli influençant mes réponses émotionnelles ?

— Le contexte de l'influence : la sémantique d'une influence contient une idée de possibilité, de probabilité. On se demande donc dans quel contexte, où et quand l'influence prend effet. Cela correspond à se demander ce qui influence une telle influence, ce qui n'est d'ailleurs pas directement représentable dans une carte cognitive. On pourrait donc se demander où, quand, dans quel contexte la publicité nous influence. Serait-ce plutôt dans les transports en ville, ou à la campagne ? Serait-ce plutôt en regardant la télé, ou en faisant les courses ?

Dans un but de clarification très partielle de ces ambiguïtés, les influences des cartes cognitives disposent généralement des trois attributs suivants [Axe15 ; Cha10] :

— La direction, représentée par l'orientation des arcs, nous permet de différencier A influence B et B influence A. A et B peuvent s'influencer mutuellement auquel cas il y a deux influences ayant une direction opposée.

— La force, représentée au travers d'un ensemble de valeurs comme {peu, modérée, beaucoup} ou {1,2,3,4}, est l'intensité de l'influence, donnant une idée de l'ampleur de la modification rendue par l'influence. La force d'une influence est indiquée par une sorte d'échelle représentée par l'ensemble de valeurs.

— La polarité, généralement représentée par le signe des valeurs d'influence, caractérise l'effet de l'influence qui agit soit positivement soit négativement. La polarité peut être particulièrement ambiguë lorsque l'on la confond avec une autre signification du terme « positif » qui serait alors compris comme « bon », « attendu » ou « espéré ». En effet, la formulation « *La vieillesse influence positivement la mort* » peut rendre ambiguë la

nature de l'influence : Est-ce-que le vieillesse entraîne une mort positive, agréable, ou est-ce-que la vieillesse augmente la probabilité de mourir ?

Les influences peuvent avoir d'autres attributs en fonction des modèles, comme un temps de latence, une probabilité ou un degré de certitude.
Ces attributs donnent des indications générales sur les influences tout en préservant une certaine souplesse, et une certaine ambiguïté.

Inférence

Depuis la formalisation d'Axelrod, les cartes cognitives permettent d'inférer l'influence propagée d'un concept sur un autre. Celle-ci représente l'influence que n'importe quel concept d'une carte a sur n'importe quel autre concept de la carte. Elle est calculée en prenant en compte tous les chemins allant du premier concept vers le second. Ensuite une valeur est attribuée à chacun de ces chemins, c'est l'influence propagée du chemin, calculée grâce à un opérateur qui dépend de l'ensemble de valeurs sur lequel est définie la carte cognitive. Les influences propagées des chemins sont par la suite agrégées grâce à un deuxième opérateur dépendant lui aussi de l'ensemble de valeurs : Ce calcul donne l'influence propagée du premier concept vers le deuxième.

Un chemin d'influence d'un concept source à un concept destination est une séquence de couples de concepts basée sur les arcs d'une carte cognitive.

Définition 4 (Chemin d'influence)**.** *Soit $CM = (N, A, labelN, labelE)$ une carte cognitive. Soient c_A et c_B deux concepts de CM. On appelle un* chemin d'influence P *de c_A vers c_B une séquence de longueur $k > 0$ d'influences $(n_i, n_{i+1}) \in A$ avec $i \in [0; k-1]$ telle que $labelN(n_0) = c_A$ et $labelN(c_k) = c_B$. On note un tel chemin : $c_A \rightarrow labelN(n_1) \rightarrow ... \rightarrow labelN(n_{k-1}) \rightarrow c_B$. c_A et c_B sont respectivement appelés concept source et concept destination, notés $source_P$ et $dest_P$. On note $length_P$ la longueur du chemin P.*

Exemple 5. *Dans la carte cognitive de la figure 1.3,*
`RégimeAutoritaire` → `ContrôleDeLaPresse` → `InformationsProRégime` *est un chemin de longueur 2 partant du concept source* `RégimeAutoritaire` *et allant vers le concept destination* `InformationsProRégime`*.*

Les cartes cognitives contiennent potentiellement des circuits, et donc dans ce cas une infinité de chemins d'influence. Pour permettre d'effectuer les calculs d'influence propagée, il est préférable de contraindre l'ensemble des chemins pris en compte : on appelle ces chemins des chemins minimaux. Il existe plusieurs définitions de chemins minimaux [LeD13], certains n'acceptent aucune répétition d'influence dans la séquence du chemin (chemins simples), d'autres

ne doivent pas passer deux fois par le même concept (chemins élémentaires). Il peut également être pertinent d'être plus flexible sur la contrainte pour prendre en compte des effets « boule de neige » par exemple en acceptant de repasser au plus 2 fois par le même concept. On considère ici qu'un chemin minimal est un chemin ne passant pas deux fois par le même concept.

Définition 5 (Chemin minimal)**.** *Soit $CM = (N, A, labelN, labelE)$ une carte cognitive. Soit P un chemin d'influence dans CM de longueur k composé des influences (n_i, n_{i+1}) avec $i < k$. P est un* chemin minimal *si et seulement si $\forall i, j < k, i \neq j \Rightarrow n_i \neq n_j \wedge n_{i+1} \neq n_{j+1}$.*
On note P_{c_1,c_2} l'ensemble des chemins minimaux entre le noeud étiqueté par le concept c_1 et le noeud étiqueté par le concept c_2.
On note $Paths_{CM}$ l'ensemble des chemins minimaux de la carte CM.

Exemple 6. *La carte cognitive (figure 1.3) contient plusieurs circuits. C'est le cas du chemin :* `RegimeAutoritaire` → `ControleDeLaPresse` → `InformationsProRegime` → `RegimeAutoritaire` *qui passe deux fois par le concept :* `RegimeAutoritaire`*. Il y a 7 chemins minimaux entre* `RegimeAutoritaire` *et* `ContestationDuRégime`*, comme* p_1=`RegimeAutoritaire` → `Repression` → `ContestationDuRégime` *ou* p_2=`RegimeAutoritaire` → `FiltrageInternet` → `LiberteDExpression` → `ContestationDuRegime`*. On a ainsi* $P_{RegimeAutoritaire,ContestationDuRegime} = \{p_1, p_2, p_3, p_4, p_5, p_6, p_7\}$.

Une fois les chemins minimaux entre un concept et un autre identifiés, pour calculer l'influence propagée il est nécessaire d'attribuer une valeur à chaque chemin, c'est l'influence propagée de chemin. Ce calcul dépend de l'ensemble de valeurs, l'ensemble choisi dans cette partie est l'ensemble de valeurs défini par Axelrod : {+,-}.

Définition 6 (Influence propagée de chemin)**.** *Soit $CM = (N, A, labelN, labelE)$ une carte cognitive définie sur l'ensemble de valeurs $I = \{+, -\}$. Soit P un chemin d'influence de longueur k dans CM composé des influences (n_i, n_{i+1}) avec $i < k$. L'influence propagée de P est :*

$$\mathcal{IP}(P) = \bigwedge_{i=0}^{k-1} labelE((n_i, n_{i+1})) \quad avec$$

$\wedge$	$+$	$-$
$+$	$+$	$-$
$-$	$-$	$+$

Exemple 7. *Considérons, dans la carte cognitive de la figure 1.3, les chemins p_1 et p_2 de l'exemple précédent. On peut ainsi calculer :*
$\mathcal{IP}(p_1) = + \wedge + = +$ *et* $\mathcal{IP}(p_2) = + \wedge - \wedge + = -$

La dernière étape du calcul de l'influence propagée d'un concept sur un autre consiste à agréger les différentes valeurs d'influence propagée de chemin obtenues grâce au calcul précédent. Cette agrégation dépend donc aussi de l'ensemble de valeurs choisi et il est possible

d'opter pour différentes manières d'agréger ces valeurs. Il est ici choisi de reprendre l'opérateur d'agrégation d'Axelrod qui attribue une influence propagée positive (+) lorsque tous les chemins sont positifs, négative (-) lorsque tous les chemins sont négatifs, ambiguë (?) lorsque il y a des chemins positifs et négatifs et nulle (0) lorsqu'il n'y a pas de chemins.

Définition 7 (Influence propagée)**.** *Soit $CM = (N, A, labelN, labelE)$ une carte cognitive définie sur l'ensemble de valeurs $I = \{+, -\}$. Soient $c_1, c_2 \in C$ deux concepts de CM. L'influence propagée de c_1 vers c_2 est une fonction $\mathcal{I} : C \times C \rightarrow \{+, -, ?, 0\}$ telle que :*

$$\mathcal{I}(c_1, c_2) = \begin{cases} 0 & si\ P_{c_1,c_2} = \varnothing \\ \bigvee\limits_{P \in P_{c_1,c_2}} \mathcal{IP}(P) & sinon. \end{cases} \qquad avec \quad \begin{array}{|c|cc|} \hline \vee & + & - \\ \hline + & + & ? \\ - & ? & - \\ \hline \end{array}$$

Exemple 8. *Cet exemple illustre les inférences d'Axelrod sur la carte cognitive de la figure 1.3. Tentons de calculer l'influence propagée de régime autoritaire vers la contestation du régime, ces deux concepts ne s'influencent pas directement. La première étape est d'identifier tous les chemins partant du concept source* `Régime autoritaire` *et allant au concept destination* `Contestation du régime`*. Nous avons vu dans les exemples précédents qu'il y en avait 7, comme le chemin* `Régime autoritaire` $\rightarrow$ `Répression` $\rightarrow$ `Contestation du régime`*. Une fois les chemins identifiés, il faut calculer la valeur d'influence propagée de chaque chemin. En utilisant l'opérateur $\wedge$ d'Axelrod on obtient les 7 valeurs : {-,-,-,-,-,-,+}. Afin de calculer l'influence propagée globale de* `Régime autoritaire` *sur* `Contestation du régime`*, le second opérateur d'Axelrod $\vee$ nous permet d'agréger les 7 valeurs d'influences propagée de chemin, ce qui nous permet d'obtenir $\mathcal{I}$(Régime autoritaire, Contestation du régime)=$- \vee - \vee - \vee - \vee - \vee - \vee + = ?$. Cela signifie qu'il y a une incertitude, l'influence propagée de* `Régime autoritaire` *sur* `Contestation du régime` *est en effet complexe car elle est à la fois positive et négative. On pourrait cependant considérer un autre opérateur qui nous donnerait un résultat différent.*

Étudions comment se situe ce modèle par rapport aux critères de comparaison.

Types de connaissances Les connaissances représentées par les cartes cognitives sont des influences entre concepts, elles décrivent un système d'influence d'un domaine particulier. Les connaissances représentées sont de natures spécifiques, réduisant l'expressivité du modèle pour se focaliser sur un type particulier de connaissance, en l'occurrence des influences.

Objectifs et applications Ce modèle permet de représenter des systèmes d'influence complexes provenant de la perception d'un ou de plusieurs individus ainsi que d'effectuer des taches d'inférence simples comme l'influence propagée. Cela rend le modèle utile à la l'aide à la prise de décision et l'étude de systèmes complexes et des croyances d'individus.

Construction et utilisation La construction d'une carte cognitive est en général effectuée au cour d'entretiens avec des personnes dont on souhaite étudier les perceptions ou qui sont experts d'un domaine d'intérêt. Elle peut être effectuée sur feuille, tableau ou bien à l'aide d'un

logiciel informatique comme MentalModeler[3]. Sa construction reste relativement aisée mais peut se complexifier dans le cas où l'on souhaite construire plusieurs cartes avec différentes personnes. Des méthodologies de construction sont présentes dans la littérature scientifique pour obtenir des cartes de qualité.

Aspect visuel et sémantique Le côté visuel des cartes cognitives est très important, que ce soit durant la construction en s'appuyant visuellement sur une carte pour l'élaborer collectivement ou durant l'analyse en s'en servant de support de discussion, notamment pour l'aide à la prise de décision. La sémantique du modèle des cartes cognitives offre une certaine flexibilité, notamment due à l'ambiguïté des influences, elle est cependant assez définie pour permettre des raisonnements simples comme l'influence propagée.

1.1.4 Cartes conceptuelles

Les cartes conceptuelles [GN84] font partie de la cartographie sémantique, elles permettent d'organiser des connaissances sous forme de graphe afin de répondre à une question centrale. Leur objectif principal est l'aide à l'apprentissage. Les cartes conceptuelles ont été inventées par le biologiste Joseph Novak, ses recherches portaient sur l'évolution du savoir scientifique des enfants. En partant de l'hypothèse que l'apprentissage fonctionne grâce à l'assimilation de nouveaux concepts et de nouvelles propositions au sein de la structure cognitive de l'apprenant, il en déduit que les cartes conceptuelles jouent un rôle majeur pour représenter la compréhension des enfants et en suivre l'évolution [GN84]. Dans un autre article [NC08], Joseph Novak formalise les cartes conceptuelles et propose une méthode générique pour la construction de ces cartes. Les cartes conceptuelles ne sont plus seulement utilisées pour des enfants mais plus généralement pour des apprenants, des chercheurs [BPL03] les ont par exemple utilisées pour des étudiants adultes dans un contexte de cours universitaire en ligne.

Sémantique

Une carte conceptuelle est un graphe orienté composé de sommets concept, de sommets relation et d'arcs reliant sommets concept et relation. Deux concepts reliés au travers d'une relation forment une proposition qui exprime la relation entre le premier concept dit « sujet » et le second concept dit « objet ».

Définition 8 (Carte conceptuelle)**.** *Soient C un ensemble de concepts et R un ensemble de relations. Une carte conceptuelle définie sur C et R est un graphe biparti orienté étiqueté CM = (V_C, V_R, A, label) tel que :*

— V_C *est l'ensemble de sommets concept.*

— V_R *est l'ensemble de sommets relation.*

3. http://www.mentalmodeler.org/

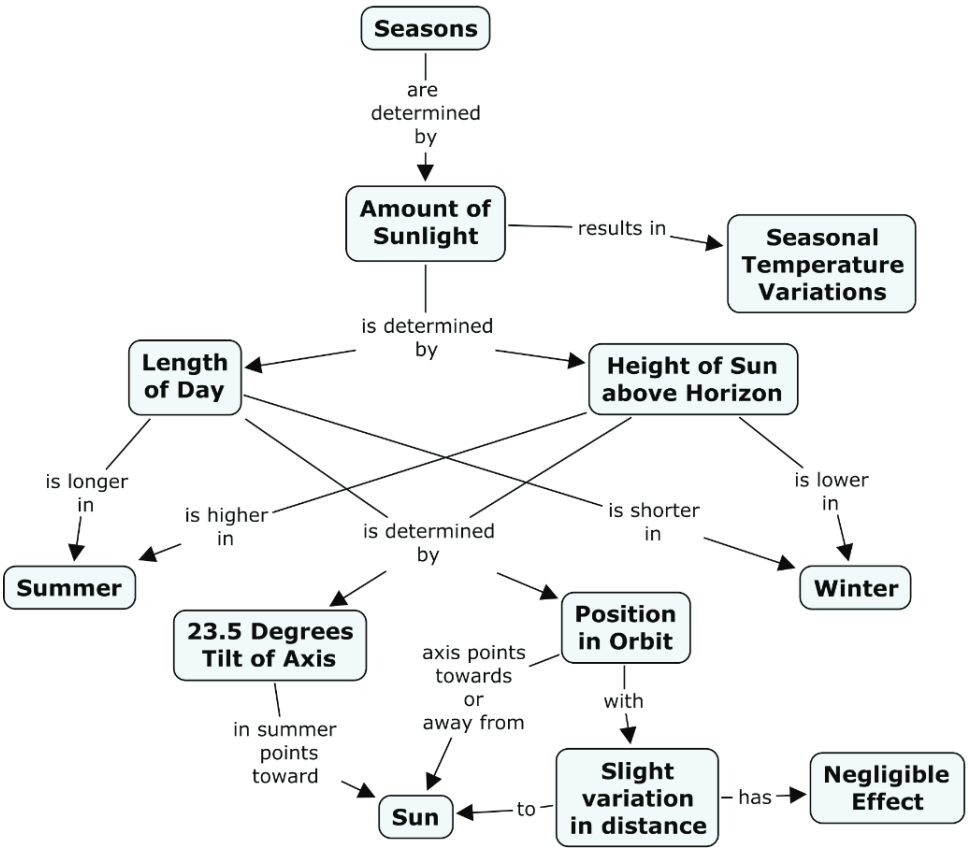


FIGURE 1.4 – Exemple de carte conceptuelle représentant les connaissances sur les saisons[NC08].

— $A \subseteq (V_C \times V_R) \cup (V_R \times V_C)$ *est l'ensemble des arcs du graphe.*

— *label est une fonction d'étiquetage des sommets attribuant à chaque sommet concept dans* V_C *un concept de C et à chaque sommet relation dans* V_R *une relation de R.*

Exemple 9. *Cet exemple se base sur la carte conceptuelle de la figure 1.4 qui provient de l'article [NC08]. Une question centrale est souvent utilisée pour bien définir le contexte et l'objectif de la carte, cette carte conceptuelle part de la question : « Pourquoi y a t-il des saisons ? ». Les sommets concept sont encadrés et étiquetés par un texte bref en gras représentant un concept, par exemple le sommet le plus haut du graphe est un sommet concept étiqueté par le concept 'Summer', c'est d'ailleurs le concept principal de la carte. Les sommets relation ne sont pas encadrés et sont étiquetés par un texte bref représentant une relation, par exemple le sommet étiqueté par la relation 'is determined by' est un sommet relation. On remarque que les arcs sont orientés et qu'ils relient soit un sommet concept à un sommet relation ('Summer' vers 'is determined by') soit un sommet relation à un sommet concept ('is determined by' vers 'amount of sunlight'). Visuellement les arcs allant d'un sommet concept à un sommet relation*

n'ont ici pas de flèche, mais sont implicitement orientés dans la direction du sommet relation.

Cette carte conceptuelle a été utilisée pour enseigner le fonctionnement des saisons à 23 étudiants de l'université d'Harvard, 21 d'entre eux n'avaient pas pu répondre à la question principale : « Pourquoi y a t-il des saisons ? ». Les cartes conceptuelles ont étés utiles pour clarifier ces connaissances et aider les étudiants à modifier leurs a priori, elles sont un outil important pour faire face aux fausses idées [Nov02].

Construction

Pour construire une carte conceptuelle, la première étape consiste à clarifier le contexte en se posant une question centrale à laquelle la carte devra répondre. Cela permet au concepteur de la carte de rester dans les limites de son sujet. La seconde étape consiste à énumérer les concepts qui seront utilisés dans la carte, sans se préoccuper de leurs relations, ni de leur emplacement final dans la carte. Ils peuvent tout de même être ordonnés en plaçant les concepts plus généraux en haut, notamment le concept central découlant de la question centrale qui est habituellement placé au dessus des autres. Pour que les apprenants puissent tirer profit de cette méthode, ceux-ci doivent construire eux-mêmes la carte conceptuelle en partant d'une ébauche de carte réalisée par le professeur. La troisième étape consiste à construire la carte en liant les concepts entre eux par des flèches via leurs relations et des mots de liaison. La dernière étape consiste à vérifier que la carte créée répond bien à la question centrale quitte à apporter des modifications ou à rajouter des concepts supplémentaires. Cette dernière étape, qui peut être effectuée en groupe, est en effet cruciale pour l'apprentissage et pour obtenir une carte conceptuelle de qualité. Il est recommandé d'utiliser soit des post-its soit un logiciel pour pouvoir modifier aisément la carte. Il existe plusieurs logiciels dédiés aux cartes conceptuelles comme VUE [4] ou CMapTools [5], mais il est également possible d'utiliser un des très nombreux éditeurs de graphe. Joseph Novak décrit en détail les méthodologies de construction ainsi que les problèmes éventuels dans son article [NC08].

Étudions comment se situe ce modèle par rapport aux critères de comparaison.

Types de connaissances Les cartes conceptuelles représentent des connaissances générales, elles sont très expressives et permettent de décrire un domaine particulier.

Objectifs et applications L'objectif de ce modèle est l'aide à l'apprentissage, il est donc utilisé comme outil pédagogique dans le domaine de l'enseignement. Le côté applicatif de ce modèle est assez restreint du fait de son objectif spécifique.

4. `https://vue.tufts.edu/`
5. `http://cmap.ihmc.us/`

Construction et utilisation La construction d'une carte conceptuelle est en général relativement aisée, il est possible de s'appuyer sur des méthodologies existantes comme celles citées précédemment. La construction peut être effectuée sur des supports variés comme un tableau, des post-its ou des logiciels dédiés.

Aspect visuel et sémantique L'aspect visuel des cartes conceptuelles est très important, puisqu'il est au coeur du processus d'apprentissage, ce qui est l'objectif du modèle. Les cartes conceptuelles disposent même de normes visuelles comme le fait de placer les concepts les plus importants en haut de la carte. La sémantique des concepts est néanmoins peu précise, afin d'apporter une flexibilité dans l'utilisation. La sémantique étant peu définie, il n'est pas possible d'effectuer de raisonnement avec ce modèle.

1.1.5 Cartes mentales

Les cartes mentales [BB93], autrement appelées cartes heuristiques, permettent de décrire une idée principale en la liant à des concepts pouvant eux-même être liés à d'autres concepts jusqu'à atteindre le niveau de description souhaité. Les cartes mentales existent depuis des siècles mais ont été popularisées par le psychologue anglais Tony Buzan dans les années 70. Sans les nommer, il les utilisa dans son projet d'encyclopédie du cerveau et les introduisit lors de cours diffusés par la BBC. Il les formalisa en 1993 et les nomma *mind maps*. Depuis, les cartes mentales se sont grandement popularisées pour de nombreuses applications. La plus fréquente est la prise de note : lors d'une telle pratique les cartes mentales s'avèrent être efficaces pour l'aide à la mémorisation, cela est dû à l'effort conceptuel nécessaire pour retranscrire un cours ou un discours en carte mentale qui n'est pas si grand mais suffisant pour la mémorisation. Les cartes mentales peuvent également être utiles lors de brainstorming, en tant que support elles favorisent l'organisation et l'échange des idées prononcées lors des discussions.

Sémantique

Une carte mentale est une représentation graphique simple, elle se présente sous la forme d'une arborescence dont la racine est le concept principal situé au centre. La racine est connectée à d'autres sommets étiquetés par des concepts plus spécifiques décrivant le concept principal. La structure radiale et arborescente d'une carte mentale exprime implicitement un ordre de priorité entre les idées par rapport à l'idée centrale. Les cartes mentales sont personnalisables et offrent beaucoup de liberté, il est ainsi possible d'étiqueter un sommet par des liens ou des images, et une arête par un ensemble d'attributs de type graphique comme la couleur, l'épaisseur ou la forme.

FIGURE 1.5 – Exemple de carte mentale représentant les bonnes pratiques de construction

Exemple 10. *Cet exemple se base sur la carte mentale de la figure 1.5*[6]*. Cette carte mentale représente les règles de bonnes pratiques pour l'utilisation de ce modèle, elle pourrait être le résultat de prise de note lors d'un exposé sur les cartes mentales. L'idée principale, « Mind map guidelines », figure au centre et elle est décrite par les différentes sections de l'arborescence mises en évidence par leur position et couleur. On notera que deux sommets, étiquetés par 'color' et 'links', sont liés par un arc alors qu'il ne font pas partie de la même branche de l'arborescence. Cela illustre la flexibilité de ce modèle qui est très facile à utiliser et peu contraignant.*

Construction

Les cartes mentales ont été formalisées au travers d'un ensemble de règles de bonne pratique concernant la construction et l'utilisation. Tony Buzan propose 10 règles pour construire une bonne carte mentale [BB06] :

— Commencer avec une image au milieu de la page en utilisant au minimum 3 couleurs.

— Utiliser des images ou des symboles partout dans la carte mentale.

6. https://en.wikipedia.org/wiki/Mind_map#/media/File:MindMapGuidlines.svg

— Choisir des mots-clés et les écrire en lettres capitales ou minuscules.
— Écrire chaque mot ou image seul sur sa propre ligne.
— Connecter les lignes, en partant de l'image centrale : les lignes proches du centre étant plus épaisses, celles en périphérie plus fines.
— Donner la même longueur aux lignes que le mot/l'image.
— Utiliser des couleurs pour l'aide à la mémoire d'évocation et stimuler la créativité.
— Développer son propre style de carte mentale.
— Surligner et montrer les associations dans la carte mentale.
— Garder la carte mentale claire en utilisant une hiérarchie radiale, un ordre numérique ou des cadres sur les branches.

Avec la popularisation des cartes mentales de nombreuses autres ressources ont vu le jour, mais les règles de Buzan restent une référence. Une carte mentale peut être dessinée par un individu sur une simple feuille de papier. Dans le cas d'une construction collective, une carte peut aussi être dessinée sur un tableau ou avec des post-its. Une carte mentale peut également être construite à l'aide d'un logiciel destiné à cet effet, il en existe beaucoup, tels que Xmind [7] ou FreeMind [8] qui est le plus connu.

Étudions comment se situe ce modèle par rapport aux critères de comparaison.

Types de connaissances Les cartes mentales représentent des connaissances générales, elles sont très expressives et permettent de décrire un domaine particulier.

Objectifs et applications L'objectif de ce modèle est l'aide à la mémorisation et la prise de notes, il est donc utilisé comme outil lors de réunions ou de cours.

Construction et utilisation La construction d'une carte mentale est particulièrement aisée, il n'est pas nécessaire de s'appuyer sur des méthodologies existantes comme les autres modèle puisque celui-ci offre beaucoup de liberté est les « bonnes pratiques » dépendent des individus. La construction peut être effectuée sur des supports papier, un tableau ou des logiciels dédiés.

Aspect visuel et sémantique L'aspect visuel des cartes mentales est très important, ce modèle se résume presque entièrement à son aspect visuel. Il est possible d'y intégrer des images, des couleurs des dessins, mais cela rend la sémantique imprécise et donc tout type de raisonnement impossible.

1.1.6 Bilan

Les cinq modèles présentés ont en commun l'utilisation d'une structure de graphe pour représenter des connaissances. Ils sont cependant très différents, que se soit dans leurs ob-

7. `https://www.xmind.net`
8. `http://freemind.sourceforge.net/wiki/index.php`

jectifs, leur utilisation, leur sémantique ou leur formalisme. Cette partie fait un bilan de comparaison entre les cinq modèles présentés, d'abord concernant les types de connaissances représentées, puis leur utilisation et leurs applications, ensuite leur construction et enfin concernant l'importance de l'aspect visuel et de la sémantique. Chacune de ces comparaisons est accompagnée d'un tableau de comparaison dont les valeurs (+,++ et +++) donnent une idée générale de l'aptitude des différents modèles.

Types de connaissances représentées

Les cinq modèles présentés, ne cherchent pas tous à représenter les mêmes types de connaissances. En effet, les logiques de descriptions, les cartes conceptuelles et les cartes mentales représentent des connaissances générales et sont plus expressives que les modèles de réseau bayésien et de carte cognitive qui représentent des connaissances d'un type particulier. Ce tableau illustre les différences parmi les connaissances représentées dans les différents modèles.

	Logique de description	*Réseau bayésien*	*Carte cognitive*	*Carte conceptuelle*	*Carte mentale*
Connaissances générales	$+++$	$+$	$+$	$+++$	$+++$
Probabilités conditionnelles	$+$	$+++$	$++$	$+$	$+$
Systèmes d'influences	$++$	$++$	$+++$	$+$	$+$

Alors que les réseaux bayésiens et les cartes cognitives représentent des connaissances spécifiques, respectivement des probabilités conditionnelles et des influences, les autres modèles présentés représentent des connaissances plus générales, comme la description d'un domaine particulier. Ces connaissances générales sont cependant représentées différemment selon les modèles, certaines sont décrites avec précision comme avec les logiques de description, d'autres sont imprécises et visuelles comme dans une carte mentale. Ces différences dépendent fortement des objectifs des modèles qui sont bien distincts.

Objectifs et applications

Chacun des cinq modèles a été défini pour des objectifs particuliers. Les logiques de description visent à décrire et raisonner, les réseaux bayésiens à prédire, les cartes cognitives sont une aide à la prise de décision et à l'étude de systèmes complexes, les cartes conceptuelles visent à répondre à une question centrale pour l'apprentissage et les cartes mentales décrivent

une idée de manière à faciliter la mémorisation. Ce tableau illustre les différences des objectifs des différents modèles.

	Logique de description	*Réseau bayésien*	*Carte cognitive*	*Carte conceptuelle*	*Carte mentale*
Raisonner	+ + +	+ + +	++	+	+
Prédire	+	+ + +	++	+	+
Aide à la décision	+	++	+ + +	+	++
Aide à l'apprentissage	+	+	+	+ + +	++
Aide à la mémorisation	+	+	++	++	+ + +

Ces modèles n'ont pas les mêmes objectifs de raisonnement. Les logiques de description servent à décrire avec précision un domaine particulier et effectuer des tâches de raisonnement. Les réseaux bayésiens et les cartes cognitives permettent également de raisonner mais de manières différentes : les réseaux bayésiens peuvent inférer des probabilités et les cartes cognitives peuvent inférer des valeurs d'influence alors que les logiques de description permettent d'effectuer des inférences plus générales sur des connaissances variées comme le fait d'inférer de nouvelles descriptions d'instances ou vérifier la consistance d'une base de connaissances. Les cartes conceptuelles et les cartes mentales ne permettent pas de raisonner.

La représentation de connaissances peut aussi permettre d'effectuer des prédictions. Les réseaux bayésiens ont pour objectif de prédire en utilisant des inférences probabilistes. Cet objectif peut être envisagé avec les cartes cognitives, notamment les cartes cognitives floues qui permettent l'exécution de scénarios pour prédire des influences, mais n'est pas envisageable avec les autres modèles présentés.

Concernant l'aide à la prise de décision, les cartes cognitives sont très adaptées en permettant l'étude de systèmes d'influences complexes. D'autres modèles peuvent être utiles pour prendre une décision, comme les cartes mentales qui peuvent être un support de discussions ou les réseaux bayésiens qui pourraient aider grâce à leurs prédictions.

Concernant l'aide à l'apprentissage, les cartes conceptuelles sont très adaptées, les cartes mentales peuvent également être utiles. Les autres modèles ne sont pas adaptés pour répondre aux attentes particulières de cet objectif.

Représenter des connaissances peut aider à mémoriser ces connaissances. C'est l'objectif principal des cartes mentales, leur simplicité, leur souplesse et leur coté visuel en font un modèle très adapté. Les cartes conceptuelles et les cartes cognitives peuvent aider visuellement

pour ces taches, mais les modèles provenant de l'IA seraient inadaptés.

Ayant des objectifs différents et ne représentant pas forcément les mêmes types de connaissances, ces modèles se construisent et s'utilisent de manières diverses.

Construction et utilisation

La construction, c'est-à-dire le processus d'acquisition et de modélisation des connaissances, est un critère important, d'une part car un modèle trop difficile à construire rend le modèle inutilisable et donc inutile, d'autre part car la méthode de construction a une influence directe sur la validité du contenu obtenu. Ce deuxième point est particulièrement important quand ces modèles sont utilisés dans un cadre scientifique puisque la méthode de construction et donc d'obtention des données doit s'inscrire dans une démarche scientifique, ce qui permet par la suite de déterminer quelle est la fiabilité des résultats. Ce tableau illustre les différences concernant la facilité de construction des différents modèles ainsi que la qualité de leurs méthodologies.

	Logique de description	*Réseau bayésien*	*Carte cognitive*	*Carte conceptuelle*	*Carte mentale*
Facilité de construction	+	+	++	++	+++
Méthodologie claire	+++	+++	++	++	+

Il semble normal qu'un modèle plus formel et contraignant soit plus difficile à construire. Ainsi les cartes mentales sont les plus faciles à construire, les cartes conceptuelles et les cartes cognitives demandent plus d'effort et de temps de construction mais restent relativement simples à construire. Les réseaux bayésiens et les logiques de description demandent un effort de construction important. Cela n'est pas si dérangeant étant donné que les grandes capacités de ces modèles motivent leur construction difficile. On ne tolérerait cependant pas de prendre des notes avec la difficulté de construction d'une ontologie.

La ligne « Méthodologie claire » du tableau donne une idée de la rigueur et de la quantité de méthodologies présentes dans la littérature concernant la construction des modèles. Il existe en effet un grand nombre de publications proposant des méthodologies rigoureuses pour la construction de logiques de description [CFG03 ; DMN09 ; UK95] et de réseaux bayésiens [HG02 ; NFN00 ; SRA90], dont certaines concernent la construction automatisée. Il en existe également beaucoup concernant les cartes cognitives [All96 ; EA13 ; ÖÖ04] et les cartes conceptuelles [NC08], bien qu'elles soient moins rigoureuses. Concernant les cartes mentales, la méthode de construction n'a pas beaucoup évolué depuis les 10 conseils de Tony Buzan vus précédemment.

Aspect visuel et sémantique

Chacun de ces modèles représente des connaissances grâce à une structure de graphe, cette structure a une sémantique particulière et elle est particulièrement adaptée à une représentation graphique. Un graphe est en effet un outil de visualisation très puissant [HMM00 ; Liu+14], qui requiert que peu d'effort d'interprétation [Zha91]. Il se trouve être également préférable aux codes visuels classiques comme la proximité ou les alignements pour représenter des relations [PR94]. Cet aspect visuel n'est cependant pas considéré avec autant d'importance dans les différents modèles présentés. Ce tableau illustre les différences quant à l'importance accordée à l'aspect visuel et à la sémantique du modèle.

	Logique de description	*Réseau bayésien*	*Carte cognitive*	*Carte conceptuelle*	*Carte mentale*
Visualisation aisée	+	++	++	++	+++
Sémantique forte	+++	+++	++	+	+

Ces modèles n'accordent pas la même importance à l'aspect visuel et à la sémantique. Les logiques de descriptions et les réseaux bayésiens ont une sémantique forte afin de permettre d'effectuer des raisonnements. Les logiques de description ne sont pas adaptées à la visualisation, les réseaux bayésiens un peu plus, mais cela est est dû au fait que les graphes de ce modèle ne représentent qu'un seul type de relation, celle de dépendance probabiliste. Les cartes conceptuelles et les cartes mentales n'ont pas une sémantique forte et attachent plus d'importance à l'aspect visuel du modèle, ne cherchant pas à effectuer de raisonnement. Les cartes cognitives accordent une importance particulière à l'aspect visuel, mais également à la sémantique, de manière à pouvoir effectuer certains raisonnements.

Positionnement des cartes cognitives

Les cartes cognitives diffèrent des autres modèles sur plusieurs points. Tout d'abord sur leurs objectifs, les cartes cognitives représentent des systèmes d'influences complexes, ce qui peut servir à analyser ces systèmes pour en comprendre le fonctionnement et éventuellement être aussi une aide à la prise de décision [Ede92]. Les connaissances représentées dans ce modèle sont très spécifiques, il s'agit d'influences. Se focaliser ainsi sur les influences et garder une expressivité restreinte rend la représentation de ces connaissances plus simple. Ces connaissances pourraient en effet être représentées avec les logiques de description ou les cartes mentales mais cela nuirait soit à la facilité de construction ou de visualisation soit à la sémantique et aux capacités de raisonnements. En ce sens, les cartes cognitives sont un

compromis entre les modèles de cartographie sémantique et de réseaux sémantiques, tout en étant spécialisées dans la représentation de systèmes d'influences.

1.2 Le modèle de carte cognitive à composante taxonomique

Le modèle de carte cognitive présenté précédemment est le modèle classique de carte cognitive. Par la suite, la plupart de la recherche en informatique s'est concentrée sur le modèle de cartes cognitives floues [Kos+86], d'autres travaux se sont basés sur le modèle classique, notamment les thèses de Lionel Chauvin [Cha10] et d'Aymeric LeDorze [LeD13]. Celles-ci ont contribué à l'amélioration du modèle classique avec des apports provenant de l'informatique. La première propose d'organiser les concepts en joignant une taxonomie de concepts au modèle. Cela permet d'avoir un vocabulaire commun entre plusieurs cartes, mais aussi d'effectuer de nouvelles opérations sur ce modèle, comme le calcul de l'influence taxonomique ou les vues de cartes cognitives. En utilisant la taxonomie, Aymeric leDorze a aussi apporté au modèle des opérations de synthèse de plusieurs cartes avec différents paramètres et a proposé des méthodes de validation de carte cognitive. Cette thèse s'inscrit dans le continuité de ces travaux, ils sont donc présentés ici. Cette partie présente d'abord ce qu'est une taxonomie, avant de définir le modèle de carte cognitive taxonomique avec l'influence propagée et l'influence taxonomique.

1.2.1 Carte cognitive taxonomique

Une taxonomie organise des concepts entre eux, elle permet d'effectuer les opérations qui vont être décrites dans cette partie comme l'influence taxonomique. La taxonomie fait partie de la méthodologie de construction de cartes cognitives [ÖÖ04], elle garantit un vocabulaire commun entre plusieurs cartes et facilite la construction. Formellement, une taxonomie associe à un ensemble de concepts une relation d'ordre partiel. La relation d'ordre partiel est une relation de spécialisation, elle peut être interprétée comme étant une relation « est une sorte de ». Une taxonomie peut être représentée par un ensemble d'arborescences.

Définition 9 (Taxonomie)**.** *Soit C un ensemble de concepts. Une* taxonomie $T = (C, \leqslant)$ *est un ensemble d'arborescences de concepts représentant une relation d'ordre partiel $\leqslant$.*

Exemple 11. *La figure 1.6 illustre la taxonomie T1. La relation d'ordre partiel $\leqslant$ y est représentée par des liens fléchés. On peut par exemple y lire que* `Pluie` *et* `Brouillard` *sont des sortes de* `MauvaisTemps` *ou que* `Autoroute` *est une sorte de* `Route`*.*

Nous définissons à présent quelques notions liées à la taxonomie qui nous seront utiles dans les définitions à venir. Nous définissons ainsi la notion de comparabilité. Intuitivement, deux concepts sont comparables si l'un est une sorte de l'autre en prenant en compte la transitivité de la relation. Deux concepts sont incomparables s'ils ne sont pas comparables.

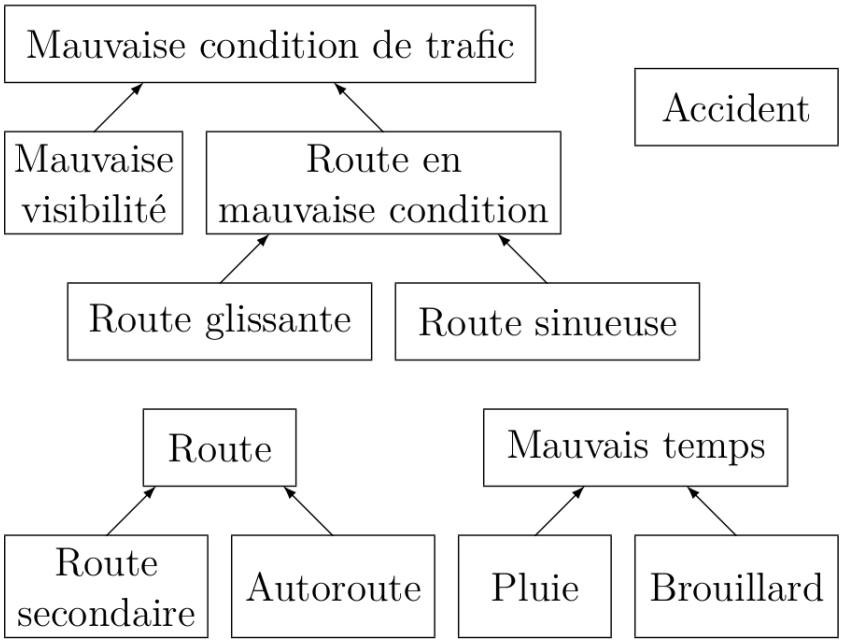


FIGURE 1.6 – Taxonomie de concepts T1.

Définition 10 (Concepts comparables)**.** *Soit $T = (C, \leqslant)$ une taxonomie. Soient $c_1, c_2 \in C$ deux concepts de la taxonomie T.*
c_1 et c_2 sont dits comparables, noté $c_1 \perp c_2$, si et seulement si $c_1 \leqslant c_2$ ou $c_2 \leqslant c_1$.
c_1 et c_2 sont dits incomparables, noté $c_1 || c_2$, si et seulement si $\neg(c_1 \perp c_2)$.

Exemple 12. *Dans la taxonomie T1 de la figure 1.6, les concepts `Pluie` est `MauvaisTemps` sont comparables, alors que `Pluie` et `Brouillard` sont incomparables. Deux concepts comparables ne sont pas forcément directement ordonnés par la taxonomie, par exemple `RouteSinueuse` est comparable avec `MauvaiseConditionDeTrafic`. Chaque concept est comparable avec lui même, d'ailleurs `Accident` n'est comparable qu'avec lui même.*

Les concepts minimaux d'un ensemble de concepts sont les concepts les plus spécifiques par rapport à l'ordre de la taxonomie. On définit de la même manière les concepts maximaux d'un ensemble de concepts comme étant les concepts les plus généraux.

Définition 11 (Concepts minimaux et maximaux)**.** *Soit $T = (C, \leqslant)$ une taxonomie. Soit $C' \subseteq C$ un ensemble de concepts.*
Les concepts minimaux de C' par rapport à T sont :
$min_T(C') = \{c \in C' | \forall c' \in C', c' \leqslant c \Rightarrow c = c'\}$
De la même manière, les concepts maximaux de C' par rapport à T sont :
$max_T(C') = \{c \in C' | \forall c' \in C', c \leqslant c' \Rightarrow c = c'\}$

Exemple 13. *Si l'on considère la taxonomie T1 de la figure 1.6 et l'ensemble de concepts* C_1*={*`Pluie, MauvaisTemps, Accident`*}, les concepts minimaux de* C_1 *par rapport à T1 sont* `Pluie` *et* `Accident`*. Les concepts maximaux de* C_1 *par rapport à T1 sont* `MauvaisTemps` *et* `Accident`*.*

Les concepts élémentaires de la taxonomie sont les concepts les plus spécialisés de la taxonomie, ce sont également les concepts minimaux de l'ensemble de tous les concepts par rapport à la taxonomie. Les concepts élémentaires pour un concept sont les concepts les plus spécialisés de ce concept.

Définition 12 (Concepts élémentaires)**.** *Soit* $T = (C, \leqslant)$ *une taxonomie de concepts et* $c \in C$*. L'ensemble des concepts élémentaires de* T *est l'ensemble* $elem(T) = min_T(C)$*.*
L'ensemble des concepts élémentaires pour c *selon* T *est un sous-ensemble de* $elem(T)$ *tel que :* $elemPour_T(c) = \{c' \in elem(T) | c' \leqslant c\}$*.*

Exemple 14. *Les concepts élémentaires de T1(figure 1.6) sont les huit feuilles de la taxonomie (*`MauvaiseVisibilité, Pluie, Accident`*...). Les concepts élémentaires pour* `MauvaiseConditionDeTrafic` *sont les trois concepts :* `MauvaiseVisibilité, RouteGlissante` *et* `RouteSinueuse`*.*

Une *carte cognitive taxonomique* [Cha10 ; CGL08] est l'association d'une carte cognitive et d'une taxonomie. Les concepts de la carte cognitive sont des concepts de la taxonomie. On appelle une carte cognitive contrainte une carte cognitive taxonomique où les concepts utilisés sont des concepts élémentaires de la taxonomie, cette propriété est requise pour certaines définitions présentées plus loin.

Définition 13 (Carte cognitive taxonomique)**.** *Une* carte cognitive taxonomique TM *définie sur un ensemble de valeurs* I *est l'association d'une taxonomie* $T = (C, \leqslant)$ *et d'une carte cognitive* $CM = (N, A, labelN, labelE)$ *définie sur* I *telle que* $\forall n \in N, labelN(n) \in C$*. Une* carte cognitive taxonomique contrainte *est une carte cognitive taxonomique telle que* $\forall n \in N, labelN(n) \in elem(T)$*.*

Exemple 15. *La figure 1.7 illustre la carte cognitive CM1. Cette carte cognitive représente un système d'influence sur le risque d'accident de la route, elle est définie sur l'ensemble de valeurs [-1 ;+1]. Cet ensemble de valeurs indique la force de l'influence, par exemple une route glissante influence plus fortement les accidents qu'une route sinueuse. Il indique aussi si l'influence agit positivement ou négativement sur le concept, par exemple* `Autoroute` *influence fortement mais négativement le fait que la route soit sinueuse, les autoroutes étant plutôt des lignes droites. La carte cognitive taxonomique TM1 est l'association de la carte cognitive CM1 et de la taxonomie T1(figure 1.6). On remarque que les concepts présents dans TM1 sont des concepts élémentaires de la taxonomie T1 ; TM1 est donc bien une carte cognitive taxonomique contrainte.*

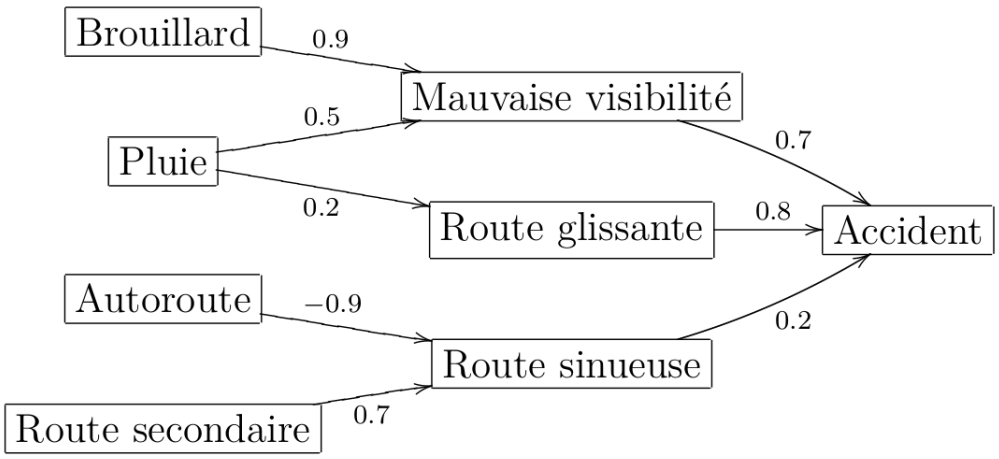


FIGURE 1.7 – Carte cognitive CM1

L'association d'une carte cognitive et d'une taxonomie de cette manière permet de définir l'opération d'influence taxonomique présentée ci-dessous, ainsi que les opérations de vue et de synthèse de cartes.

1.2.2 Influence taxonomique

L'influence propagée permet d'obtenir une indication de l'influence de n'importe quel concept de la carte cognitive vers n'importe quel autre concept. L'influence taxonomique donne une indication sur l'influence de n'importe quel concept de la taxonomie à n'importe quel autre concept. Bien que l'influence propagée ait déjà été définie précédemment, elle est ici définie pour l'ensemble de valeurs [-1 ;1], les calculs d'influence d'influence taxonomique étant définis sur cet ensemble, comme décrit par Lionel Chauvin [Cha10].

Influence propagée sur [-1 ;1]

Une fois les chemins minimaux entre un concept et un autre identifiés, pour calculer l'influence propagée il est nécessaire d'attribuer une valeur à chaque chemin, c'est l'influence propagée de chemin. Ce calcul dépend de l'ensemble de valeurs, l'ensemble choisi dans ce chapitre est $\mathcal{I}$=[-1 ;1]. Pour un ensemble de valeurs particulier, il est également possible de choisir différent calculs d'influence propagée de chemin. Il est choisi ici de faire un produit des valeurs des influences composant le chemin, cela permet de généraliser le calcul proposé par Robert Axelrod [Axe15] et son ensemble de valeurs {+,-}. En associant la valeur + à 1 et la valeur - à -1, on retrouve en effet la même influence propagée sur un chemin.

Définition 14 (Influence propagée de chemin)**.** *Soit $CM = (N, A, labelN, labelE)$ une carte cognitive définie sur l'ensemble de valeurs $I = [-1; 1]$. Soit P un chemin d'influence de lon-*

gueur k dans dans CM composé des influences (n_i, n_{i+1}) avec $i < k$. L'influence propagée de P est :

$$\mathcal{IP}(P) = \prod_{i=0}^{k-1} labelE((n_i, n_{i+1}))$$

Exemple 16. *Considérons, dans la carte cognitive CM1(figure 1.7), les chemins p_1=`Pluie` → `MauvaiseVisibilité` → `Accident` et p_2= `Pluie` → `RouteGlissante` → `Accident`. En faisant le produit on obtient $\mathcal{IP}(p_1) = 0.5 \times 0.7 = 0.35$ et $\mathcal{IP}(p_2) = 0.2 \times 0.8 = 0.16$.*

Les produits de valeurs entre 0 et 1, ou plutôt -1 et 1, ont tendance à tendre vers 0. Cela peut être sémantiquement pertinent, c'est-à-dire que l'on peut imaginer qu'un concept influençant un autre par chaîne de nombreuses influences n'aurait en somme qu'une influence assez faible sur le second concept. Le produit a aussi l'avantage de donner un résultat appartenant à l'ensemble de valeurs. D'un autre coté, il peut également être judicieux de choisir un calcul ne subissant pas cet affaissement des valeurs vers 0, par exemple en pondérant le produit en prenant en compte la longueur du chemin.

La dernière étape du calcul de l'influence propagée d'un concept sur un autre consiste à agréger les différentes valeurs de chemin obtenues grâce au calcul précédent. Cette agrégation dépend donc aussi de l'ensemble de valeurs choisi et il est possible d'opter pour différentes manières d'agréger ces valeurs. Il est ici choisi de faire une moyenne des valeurs d'influence propagée de chemin. Cette définition permet de quantifier la valeur ambiguë (?) qui apparaissait avec les opérateurs de R. Axelrod.

Définition 15 (Influence propagée)**.** *Soit $CM = (N, A, labelN, labelE)$ une carte cognitive définie sur l'ensemble de valeurs $I = [-1; 1]$. Soient $c_1, c_2 \in C$ deux concepts de CM. L'influence propagée de c_1 vers c_2 est une fonction $\mathcal{I} : C \times C \to [-1; 1]$ telle que :*

$$\mathcal{I}(c_1, c_2) = \begin{cases} 0 & si\ P_{c_1,c_2} = \varnothing \\ \frac{1}{|P_{c_1,c_2}|} \times \sum\limits_{P \in P_{c_1,c_2}} \mathcal{IP}(P) & sinon. \end{cases}$$

Exemple 17. *Dans la carte cognitive CM1(figure 1.7), calculons l'influence propagée de `Pluie` sur `Accident`. Nous avons déjà identifié les deux chemins minimaux de `Pluie` vers `Accident` et calculé leur valeur d'influence propagée de chemin. On a donc $\mathcal{I}(Pluie, Accident) = \frac{0.35+0.16}{2} = 0.255$, globalement la pluie a une influence faible et positive sur les accidents.*

Influence taxonomique

Le modèle de carte cognitive taxonomique, qui associe une taxonomie de concept à une carte cognitive, permet le calcul d'une influence propagée plus générale : l'influence taxonomique. Les formalismes présentés ici considèrent une carte cognitive taxonomique contrainte, offrant plus de simplicité dans les descriptions de ces opérations. L'influence taxonomique

donne une indication sur l'influence globale d'un concept de la taxonomie à un autre. C'est une agrégation des influences propagées des concepts élémentaires du premier concept vers les concepts élémentaires du second concept. Comme pour l'influence propagée, le calcul de l'influence taxonomique dépend de l'ensemble de valeurs et du type d'indication que l'on souhaite obtenir, comme par exemple une force moyenne des influences ou un intervalle de valeurs. Il est choisi ici d'agréger les valeurs en construisant le plus petit intervalle contenant toutes les valeurs, ce qui donne une indication quant aux différentes façons dont le premier concept peut influencer le second.

Définition 16 (Influence taxonomique)**.** *Soit une taxonomie $T = (C, \leqslant)$, un ensemble de valeurs $I = [-1; 1]$. Soit $TM = (N, A, labelN, labelE)$ une carte cognitive taxonomique contrainte définie sur T et I. Soit $c_1, c_2 \in C$ deux concepts de la taxonomie. L'influence taxonomique de c_1 sur c_2 est une fonction*[9] *$\mathcal{I}_T : C \times C \to 2^{[-1;1]}$ telle que :*

$$\mathcal{I}_T(c_1, c_2) = \left[\min_{\substack{c'_1 \in elemPour_T(c_1) \\ c'_2 \in elemPour_T(c_2)}} \mathcal{I}(c'_1, c'_2) \; ; \; \max_{\substack{c'_1 \in elemPour_T(c_1) \\ c'_2 \in elemPour_T(c_2)}} \mathcal{I}(c'_1, c'_2) \right]$$

Exemple 18. *Considérons la carte cognitive TM1 (figures 1.7 et 1.6). Cherchons à calculer l'influence globale de* `MauvaisTemps` *vers* `MauvaiseConditionDetrafic`*, ces deux concepts ne font pas partie de la carte mais ils généralisent des concepts en faisant partie. Les concepts élémentaires de* `MauvaisTemps` *sont* `Pluie` *et* `Brouillard`*, les concepts élémentaires de* `MauvaiseConditionDetrafic` *sont* `MauvaiseVisibilité`*,* `RouteGlissante` *et* `RouteSinueuse`*. Il faut donc maintenant calculer les influences propagées de chaque concept élémentaire de* `MauvaisTemps` *sur chaque concept élémentaire de* `MauvaiseConditionDetrafic` *:*

$\mathcal{I}(\texttt{Pluie}, \texttt{MauvaiseVisibilité}) = 0.5$
$\mathcal{I}(\texttt{Pluie}, \texttt{RouteGlissante}) = 0.2$
$\mathcal{I}(\texttt{Pluie}, \texttt{RouteSinueuse}) = 0$
$\mathcal{I}(\texttt{Brouillard}, \texttt{MauvaiseVisibilité}) = 0.9$
$\mathcal{I}(\texttt{Brouillard}, \texttt{RouteGlissante}) = 0$
$\mathcal{I}(\texttt{Brouillard}, \texttt{RouteSinueuse}) = 0$

L'influence taxonomique de `MauvaisTemps` *sur* `MauvaiseConditionDetrafic` *est donc :*
$\mathcal{I}_T(\texttt{MauvaisTemps,MauvaiseConditionDeTrafic}) = [0; 0.9]$
Il en résulte que le mauvais temps influence plus ou mois fortement mais toujours 'positivement' les accidents.

Le choix d'obtenir un intervalle est utile pour analyser les différentes façons dont un concept général peut influencer un second. Pour cette même raison, il pourrait être judicieux d'utiliser également un intervalle pour l'influence propagée classique. En effet, s'il y a trois chemins d'un concept A vers un concept B, et que ces chemins ont pour valeurs : -0.3 ; 0.8, -0.5, alors il serait dommage d'effectuer une moyenne et obtenir une influence propagée de A sur B égale à 0, ce

9. $2^{[-1;1]}$ est l'ensemble des parties de $[-1; 1]$.

qui nous indiquerait en rien la nature de l'influence entre ces deux concepts. L'intervalle [-0.5, 0.8] serait un résultat plus pertinent. Cela pour mettre en évidence que le choix des calculs pour les influences propagées dépend fortement du contexte et de ce que l'on souhaite obtenir, en plus de dépendre de l'ensemble de valeurs.

Nous avons présenté dans cette partie le modèle de cartes cognitives taxonomiques avec l'opération d'influence taxonomique. D'autres travaux ne sont pas décrits ici, notamment des travaux sur les vues de cartes cognitives qui permettent de générer une carte plus réduite et adaptée à un contexte [Cha10], sur la synthèse de cartes qui permet de regrouper différentes cartes entre elles de manière paramétrable, sur la validation de cartes cognitives [LeD13 ; LeD+13] qui permettent de s'assurer de la cohérence d'une carte et de mettre en évidence des contradictions, ou encore les cartes contextuelles qui catégorisent les influences en fonction de leur nature. D'autres travaux concernent la constrution et l'analyse de cartes cognitives, ils sont abordés dans la prochaine partie décrivant en partie les applications de ce modèle.

1.3 Application des cartes cognitives au domaine de la pêche

Cette partie consiste dans un premier temps à présenter le projet Kifanlo, qui est un projet de recherche en géographie qui a utilisé le modèle de cartes cognitives taxonomiques. Ce modèle est d'ailleurs nommé *graphes cognitifs* dans le domaine de la géographie étant donnée la sémantique du mot *carte* dans ce domaine. Ce projet a nourri les recherches de cet ouvrage et a permis de les valider. Dans un second temps, cette partie décrira la méthodologie employée concernant la construction des cartes dans le projet Kifanlo. Enfin, une dernière partie discutera du lien entre le projet Kifanlo et cette thèse.

Projet Kifanlo

Kifanlo [10] est un projet financé par la *Fondation de France* à hauteur de 400000 euros et qui est mené par des chercheurs en géographie de Nantes (Géolittomer, CNRS) et implique des chercheurs en informatique (LERIA, LINA) et des comités de pêcheurs (COREPEM). Ce projet vise à étudier les enjeux et évolutions des stratégies de pêche en Loire-Atlantique entre 1970 et 2010. Dans un contexte d'exploitation de l'espace maritime qui est de plus en plus important (Éoliennes, extraction de matières, zones protégées...) [GR05] qui implique des contraintes pour l'espace de pêche [Dup01 ; JP89], les stratégies de pêches restent très mal connues [CT+95]. C'est cela qui motiva le projet Kifanlo. Pour mener à bien ce projet, le modèle des cartes cognitives a semblé être pertinent pour trois raisons principales. Premièrement, les cartes cognitives permettent de prendre en compte tout type de facteurs, qu'ils soient éco-

10. Le Kifanlo est à un chalutier-thonier des Sables d'Olonne, premier navire de pêche a avoir été classé monument historique en 1984

nomiques, physiques, sociologies ou psychologiques, ainsi que d'obtenir une vue d'ensemble des relations que ces facteurs ont entre eux. Deuxièmement, puisqu'il n'existe pas de jeu de données homogène sur les stratégies de pêches au cours du temps, il est nécessaire d'en acquérir, et les cartes cognitives étant construites lors d'entretiens, peuvent palier ce problème. Troisièmement, leur simplicité d'utilisation permet de les exploiter avec des utilisateurs qui ne sont pas familiers avec les modèles de représentation de connaissances. Le modèle des cartes cognitives a d'ailleurs été utilisé dans des projets aux objectifs similaires concernant l'espace maritime et les pratiques de pêche [Chr11 ; Poi06 ; Pri+08].

Construction et analyses préliminaires des cartes

La première étape de construction de cartes cognitives consiste généralement à identifier les différents groupes d'acteurs, pour ensuite pouvoir étudier les différentes perceptions de ces groupes [Ede92]. Dans le cas du projet Kifanlo, deux groupes de pêcheurs ont été déterminés selon un échantillonage stratifié et une allocation proportionnelle. La stratification était basée sur les métadonnées suivantes : le port de rattachement, le matériel utilisé et la taille des bateaux. Ensuite une liste de pêcheurs a été constituée aléatoirement à partir du groupe des années 2010. Étant donné l'absence de base de données de bateaux pour les années 1970, une autre liste a simplement été constituée à partir de types de bateaux pour le second groupe des années 1970. Finalement, deux groupes de pêcheurs ont été interrogés : 30 pêcheurs à la retraite qui travaillaient en 1970 et 23 pêcheurs actifs. Cela représente approximativement 5% de l'effectif actif de 2010 et 2,4% de l'effectif actif de 1970. En détail, ces groupes ne correspondent pas exactement à la stratification effectuée, ce qui peut nuire à une analyse très poussée, mais la taille des groupes permet une analyse pertinente et la validation du modèle. En effet, au delà de 15 cartes cognitives, la quantité d'information apportée est très mince [ÖÖ04].

Une carte cognitive peut être obtenue de quatre manières différentes [ÖÖ04] :

— Depuis des questionnaires.

— Par extraction depuis des textes.

— Depuis des données mettant en évidence des relations causales

— Au travers d'entretiens avec un acteur qui les dessine directement.

Dans le cas du projet Kifanlo, les cartes cognitives ont été construites avec les pêcheurs lors d'entretiens. De manière plus générale, les entretiens avec le groupe des années 1970 ont duré quatre heures divisées en deux sessions et ont permis d'obtenir en plus des cartes cognitives, une histoire de vie professionnelle et personnelle ainsi que les zones de pêche utilisées. Chaque participant du groupe des années 1970 a reçu deux semaines avant l'entretien un corpus de documents permettant de se remémorer la situation et les enjeux du moment, ces documents étaient également utilisés pendant les entretiens. Concernant le groupe des an-

nées 2010, l'entretien a été effectué en une session qui permit d'obtenir l'histoire de vie et la carte cognitive, d'autres informations avaient déjà été récoltées pour un autre projet.

Les pêcheurs disposaient de la taxonomie lors de la construction des cartes, elle a tout de même été complétée au fur et à mesure des entretiens par les chercheurs en géographie et en informatique. Bien que les entretiens aient été menés par les géographes, les informaticiens étaient aussi impliqués lors de la phase de collecte de données pour assister sur le modèle des cartes cognitives. Ainsi les chercheurs ont modifié les cartes cognitives au cours de l'élaboration de la taxonomie à l'aide de notes écrites et d'enregistrements audio.

Afin de formaliser et valider les cartes cognitives construites par les pêcheurs, les cartes ont été éditées au sein du logiciel VSPCC. Ce logiciel a été développé au LERIA notamment lors de la thèse d'Aymeric Ledorze [LeD13] et de son contrat de post-doctorat.

En s'appuyant sur les travaux de cette thèse, il a été possible de générer plusieurs synthèses de carte partageant des métadonnées communes telles que les espèces pêchées ou les longueurs de bateaux ainsi qu'une synthèse générale de toutes les cartes. Grâce à ces cartes, les principaux facteurs influençant les choix des pêcheurs ont pu être facilement identifiés. Le mécanisme de vue [Cha10] a également été utilisé pour simplifier les cartes et permettre de les analyser. La synthèse des 53 cartes cognitives est en effet inutilisable visuellement sans les simplifications que propose le mécanisme de vues.

Lien du projet Kifanlo avec cette thèse

Cette thèse s'est effectuée en collaboration avec les chercheurs en géographie, notamment Brice Trouillet qui a co-encadré cette thèse et mené le projet Kifanlo. Cette collaboration vise d'un coté à apporter des outils bénéfiques aux géographes pour mener à bien leur projet et d'un autre coté à apporter un cas d'application aux recherches effectuées sur le modèle des cartes cognitives taxonomiques.

Ce cas d'application est bénéfique pour deux raisons. Premièrement car il apporte une source d'inspiration pertinente. En effet, rien de tel qu'un cas applicatif pour mettre en avant un manque ou des difficultés du modèle qui ont pu être rencontrées. Des besoins clairs peuvent même être exprimés par les chercheurs en géographie. C'est également une source d'inspiration car tout au long de cette thèse, l'accès à la disposition de l'ensemble des 53 cartes cognitives ainsi ainsi que le contexte dans lequel ces connaissances ont été recueillies, a permis de se mettre à la place d'un utilisateur pour imaginer des outils et méthodes adéquates pour la modélisation et l'analyse des cartes. Le second côté bénéfique est celui de la validation des travaux de recherche. Il est effectivement utile de disposer d'un retour d'expérience quant à la mise en pratique d'un nouveau modèle ou de nouvelles opérations dont dispose le modèle. Le projet Kifanlo avait déjà par le passé pu servir de validation des travaux sur les mécanismes de vue et de synthèse formalisés respectivement par Lionel Chauvin [Cha10 ; CGL09] et Aymeric LeDorze [LeD13 ; LeD+12]. Les besoins potentiellement dégagés par les chercheurs du projet Kifanlo servent aussi en soi de validation, illustrant alors un manque du modèle et l'utilité

de nouveaux formalismes qui pourraient les combler.

Conclusion

Dans ce chapitre nous avons présenté le modèle des cartes cognitives ainsi que son contexte. Dans un premier temps en le plaçant en comparaison avec d'autres modèles de représentation de connaissances à base de graphes, tels que les logiques de descriptions et les réseaux bayésiens qui sont des modèles formels de l'intelligence artificielle ou les cartes conceptuelles et les cartes mentales qui sont des modèles simples, pratiques mais ambigus de cartographie sémantiques. Le modèle de cartes cognitives est présenté comme un modèle simple de cartographie sémantique, mais suffisamment formel pour permettre des apports provenant de l'intelligence artificielle. Dans un second temps, le modèle de cartes cognitives taxonomiques sur lequel se basent les recherches de cette thèse est présenté avec les mécanismes formalisés lors de précédents travaux. Enfin le projet de recherche Kifanlo est présenté, il utilise le modèle des cartes cognitives taxonomiques, sert d'inspiration et de validation pour les recherches présentées dans cette thèse.

Le chapitre 2 concerne un modèle étendu de cartes cognitives, qui inclut le modèle des carte cognitive taxonomique, il y rajoute une composante temporelle et une composante spatiale. Ce nouveau modèle étendu de cartes cognitives se nomme *cartes cognitives ontologiques*.

CHAPITRE 2

LE MODÈLE DES CARTES COGNITIVES ONTOLOGIQUES

Introduction

Quand on construit une carte cognitive, ses concepts et ses influences ont généralement des caractéristiques spatiales et temporelles. Prenons le cas d'une carte cognitive représentant les influences pouvant mener à un accident de la route, comme celle utilisée lors du précédent chapitre pour illustrer les différentes notions du modèle. Chaque concept, comme `Route`, `Trafic` ou `Mauvaise météo` sous-entend une certaine temporalité ainsi qu'une localisation. Par exemple le trafic n'est pas généralisé partout, il a une localisation comme l'autoroute ou le périphérique, et il n'est pas constant, il peut être présent à différents moments de la journée. De la même manière, la mauvaise météo concerne un endroit à un moment où il fait mauvais et la route est à une localisation précise.

Malgré l'omniprésence des aspects temporels et spatiaux, le modèle des cartes cognitives ne propose quasiment rien pour les prendre en compte. Cela peut engendrer des ambiguïtés et empêche tout type de raisonnement concernant ces aspects spatiaux-temporels. Ce manque s'est aussi fait ressentir lors de projets applicatifs utilisant les cartes cognitives [Chr11].

Ce chapitre présente un nouveau modèle de cartes cognitives ontologiques, qui se base sur une ontologie spatio-temporelle sur laquelle est définie une carte cognitive taxonomique. L'intérêt de ce modèle réside dans sa capacité à décrire les concepts des cartes cognitives en leur associant des caractéristiques temporelles et spatiales. Pour cela, une carte cognitive taxonomique est définie sur une ontologie spatio-temporelle et les concepts des cartes peuvent être caractérisés spatialement et temporellement. La partie temporelle de l'ontologie se base sur un type d'entité temporelle particulier, les intervalles périodiques [Osm99] qui prennent en compte la périodicité des concepts et sont plus appropriés que d'autres entités comme les instants ou les dates. La partie spatiale de l'ontologie se base sur des entités spatiales génériques, couramment utilisées dans la littérature [Fra92 ; RC89]. Des relations qualitatives, appelées *prédicats de comparaison*, sont utilisées pour comparer les entités temporelles ou spatiales entre elles dans des *assertions spatio-temporelles*. Les connaissances spatio-temporelles sont donc représentées dans des assertions qui sont des triplets entité-

prédicat-entité. Ces assertions sont utilisées d'une part dans l'ontologie et d'autre part dans les cartes pour lier les noeuds à l'ontologie.

Ce chapitre présente dans une première partie la composante temporelle du modèle, définissant une ontologie temporelle. Il présente en deuxième partie la composante spatiale du modèle, en définissant de manière analogue à la composante temporelle une ontologie spatiale se basant sur des entités spatiales. La troisième partie définit le modèle de cartes cognitives ontologiques, qui définit une carte cognitive taxonomique sur une ontologie spatio-temporelle.

2.1 La composante temporelle

Les représentations du temps dans la littérature considèrent en général plusieurs types d'entités temporelles [Erm+14 ; Lad86] : des instants, des événements, des durées ou des intervalles de temps voire des intervalles de temps flous [Ea08]. L'ontologie temporelle présentée dans ce chapitre se base sur un type d'entité temporelle particulier : les intervalles périodiques[Osm99]. Les intervalles périodiques sont des intervalles de temps qui prennent en compte une certaine périodicité, ils sont particulièrement adaptés au modèle de cartes cognitives qui ne représentent pas des faits mais plutôt des informations d'un ordre plus général. Les concepts des cartes ont tendance à avoir une temporalité qui ne désigne pas de moment en particulier mais des événements récurrents, on peut illustrer ceci par le concept d'obscurité qui pourrait être présent dans la carte utilisée dans le précédent chapitre car elle influence une mauvaise visibilité : l'obscurité ne se situe pas à un instant précis, mais peut être présente lors d'intervalles de temps qui reviennent périodiquement en suivant les cycles des jours et des nuits.

Cette section présente les intervalles périodiques, puis propose des prédicats temporels de comparaison permettant de former des assertions temporelles qui décrivent des relations entre les intervalles périodiques. Une ontologie temporelle composée de ces connaissances est finalement définie.

Les intervalles périodiques

Un intervalle périodique est un type d'intervalle non-convexe [Lad86 ; Lad87], c'est-à-dire un intervalle composé de sous intervalles discontinus. Les intervalles périodiques ont la particularité d'être composés de sous intervalles ayant la même longueur et étant également espacés. Par exemple 'Hiver' est un intervalle périodique. Les intervalles périodiques de Balbiani et Osmani [BO00 ; Osm99] prennent en compte différentes relations qualitatives. Au lieu de spécifier exhaustivement chacune des propriétés des intervalles périodiques, comme les instants exacts

de début et de fin ainsi que les durées des intervalles, les intervalles périodiques peuvent alors être décrits en étant mis en relation les uns avec les autres. Ces propriétés ne sont en effet pas forcément connues quand on construit une carte cognitive. Cette approche est particulièrement adaptée au projet Kifanlo et semble, en général, adéquate pour les cartes cognitives, offrant de la flexibilité et permettant de faire face à des informations imprécises. Ces intervalles périodiques sont donc choisis comme base des entités temporelles dans cette thèse.

Définition 17 (Intervalle Périodique)**.** *Un* intervalle périodique *est un intervalle non-convexe dont les sous-intervalles sont également espacés et ont la même longueur.*

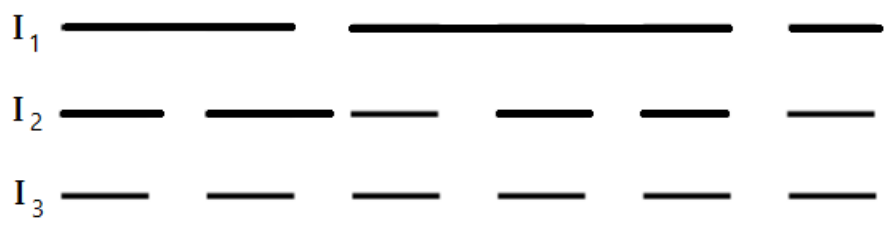


FIGURE 2.1 – Trois intervalles de temps non-convexes : I_1, I_2 et I_3.

Exemple 19. *Cet exemple se base sur la figure 2.1. Les trois intervalles de temps I_1, I_2 et I_3 sont bien des intervalles non-convexes car ils sont discontinus. On remarque que les sous-intervalles de I_1 et de I_2 ne sont pas également espacés et n'ont pas tous la même longueur, I_1 et I_2 ne sont donc pas des intervalles périodiques. I_3 en revanche a bien des sous intervalles de même longueur et également espacés, c'est un intervalle périodique.*

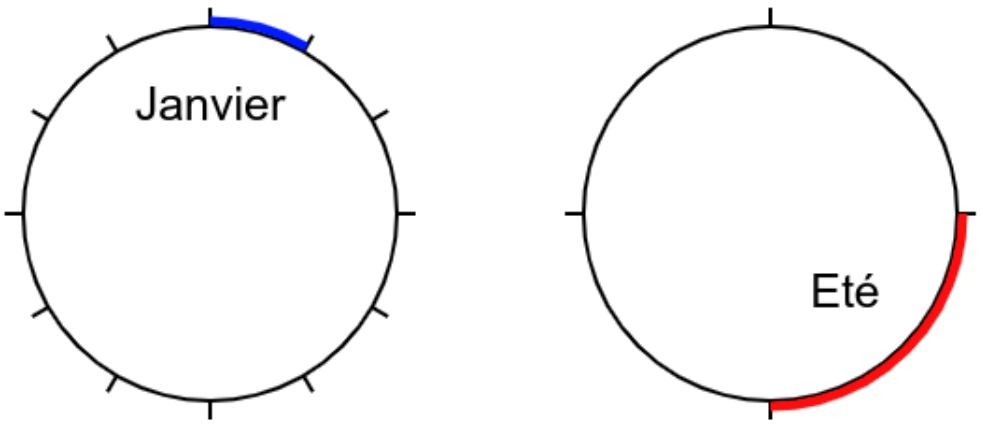


FIGURE 2.2 – Représentation cyclique des intervalles périodiques « Janvier » et « Été ».

Exemple 20. *Cet exemple se base sur la figure 2.2 qui illustre deux intervalles périodiques grâce à une représentation sous forme de cercle qui est à lire dans le sens des aiguilles d'une montre. Le premier cercle représente l'intervalle périodique* Janvier *qui prend un douzième du cercle ; la totalité du cercle représente donc la période de l'intervalle périodique, ici une année. De la même manière le second cercle représente l'intervalle périodique* Été*, sa période est également d'une année.*

Les prédicats temporels de comparaison

N'utilisant pas d'autres classes temporelles comme des dates ou des instants pour décrire les intervalles périodiques, il est proposé de les spécifier grâce à des relations qualitatives entre deux intervalles en utilisant des prédicats de comparaison. Ces prédicats sont les 16 relations de Balbiani et Osmani [Osm99] plus cinq autres relations. Les relations d'Osmani sont très similaires aux 13 relations des intervalles de James Allen [All83], remplaçant la relation de précédence et son inverse par cinq relations prenant en compte la périodicité. Ici, nous considérons aussi deux relations (Inside/Disjoint) qui combinent des relations d'Osmani et trois relations ($<_{\mathrm{T}}$, $>_{\mathrm{T}}>$, $=_{\mathrm{T}}$) qui comparent les durées des intervalles [ØH07 ; Wal47].

Définition 18 (Prédicat temporels de comparaison)**.** *Un* prédicat temporel de comparaison *est une relation binaire dont le domaine et le codomaine sont des intervalles périodiques.* $\mathcal{P}_{\mathrm{t}}$ *est l'ensemble des 21 prédicats temporels de comparaison :*

$$\{m,mi,s,si,d,di,f,fi,o,oi,eq,ppi,mmi,moi,omi,ooi,in,dis,<_{\mathrm{T}},=_{\mathrm{T}},>_{\mathrm{T}}\}$$

Le tableau ci-dessous montre les 16 relations de Balbiani et Osmani [Osm99], puis deux relations ajoutées qui généralisent certains prédicats et enfin les trois prédicats de comparaison de durées. La première colonne donne le nom du prédicat temporel de comparaison, la deuxième donne le sens du prédicat et la troisième donne la relation inverse du prédicat. Certains prédicats sont définis par leur relation inverse comme « is met by » qui est défini par le prédicat « meets », c'est pour cela qu'il n'y a que 14 prédicats dans la première colonne, 7 sont en effet définis par leur relation inverse.

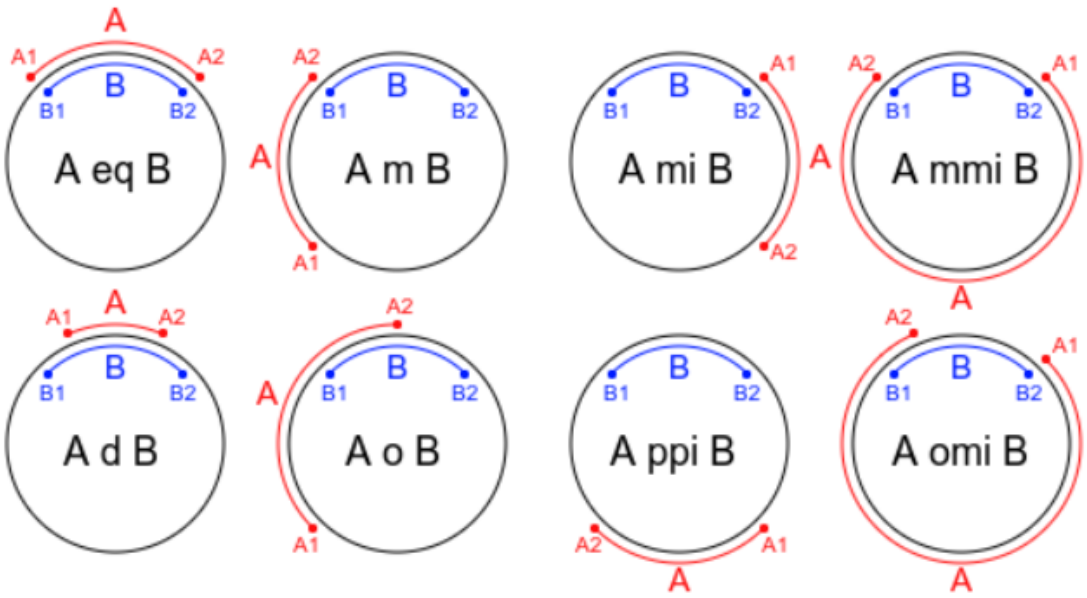


FIGURE 2.3 – Illustration de 6 prédicats temporels de comparaison.

nom	*sens*	*inverse*
eq $(equals)$	$A1 = B1\ ,\ A2 = B2$	(eq)
m $(meets)$	$A1\ ,\ A2 = B1\ ,\ B2$	**mi**
s $(starts)$	$A1 = B1\ ,\ A2\ ,\ B2$	**si**
d $(during)$	$A1\ ,\ A2\ ,\ B2\ ,\ B1$	**di**
f $(finishes)$	$A1\ ,\ A2 = B2\ ,\ B1$	**fi**
o $(overlaps)$	$A1\ ,\ B1\ ,\ A2\ ,\ B2$	**oi**
ppi $(precedes\ \&$ $is\ preceded)$	$A1\ ,\ A2\ ,\ B1\ ,\ B2$	(ppi)
mmi $(meets\ \&$ $is\ met)$	$A1 = B2\ ,\ A2 = B1$	(mmi)
moi $(meets\ \&$ $is\ overlapped)$	$A1\ ,\ B2\ ,\ A2 = B1$	**omi**
ooi $(overlaps\ \&$ $is\ overlapped)$	$A1\ ,\ B2\ ,\ B1\ ,\ A2$	(ooi)
in $(inside)$	$s \vee d \vee f \vee eq$	
dis $(disjoint)$	$m \vee mi \vee mmi \vee ppi$	
$<_{\mathbf{T}}\ (is\ shorter)$	$\overline{A1A2} < \overline{B1B2}$	$>_{\mathbf{T}}$
$=_{\mathbf{T}}\ (has\ same\ length)$	$\overline{A1A2} \simeq \overline{B1B2}$	$(=_{\mathrm{T}})$

Les 16 premiers prédicats du tableau sont les relations de Balbiani et Osmani, leur sens est renseigné grâce à la théorie CYCORD (CYClic ORDer) [Röh94] qui ordonne les bornes

A1, A2 et B1, B2 des intervalles périodiques respectifs A et B. A l'aide de la figure 2.3, expliquons le prédicat m (meets) de la deuxième ligne du tableau, son sens est donné par l'ordre (A1,A2=B1,B2), on a alors « A m B » si et seulement si cet ordre est respecté. Dans un ordre cyclique, le choix de la première borne de l'ordre est arbitraire, c'est pour cela que tous les ordres du tableau commencent par la borne inférieure du premier intervalle (A1). Les ordres (A1,A2=B1,B2), (A2=B1,B2,A1) et (B2,A1,A2=B1) sont effectivement équivalents. Expliquons l'ordre (A1,A2=B1,B2), définissant le prédicat « meets » : il signifie que si l'on commence par la borne inférieure de l'intervalle périodique A (A1), viendra ensuite la borne supérieure de A (A2) et la borne inférieure de B (B1) qui sont au même moment, puis la borne supérieure de B (B2). Dit autrement, l'intervalle B commence quand l'intervalle A finit, et se termine avant que l'intervalle A commence, comme par exemple les mois Juillet et Août. La relation inverse de « meets » est « is met by », pour obtenir l'ordre la définissant il suffit d'intervertir les intervalles A et B, ainsi mi (is met by) est définie par l'ordre (A1=B2,A2,B1) et on a par exemple « Août is met by Juillet ». La figure 2.3 illustre également d'autres prédicats, comme ppi, mmi ou omi qui sont plus complexes, afin de faciliter leur compréhension. Ces prédicats sont en effet spécifiques aux intervalles périodiques, mmi par exemple, est un prédicat signifiant que le premier intervalle rencontre le second et le second rencontre le premier, ce qui est exprimable par l'ordre (A1=B2,B1=A2).

Les deux relations ajoutées sont :'in'(Inside) qui est la disjonction de 'starts', 'during', 'finishes', 'equals' et 'dis'(Disjoint) qui est la disjonction de 'meets', 'is met by', 'meets & is met', 'precedes & is preceded'. Ces deux relations servent à spécifier des relations de manière plus générales, en l'occurrence quand un intervalle est complètement inclus dans un autre ou quand il est complètement disjoint.

Le bas du tableau concerne trois relations pour comparer les durées des intervalles périodiques : $<_{\mathrm{T}}$, $>_{\mathrm{T}}$ et $=_{\mathrm{T}}$. Leur signification est triviale, A $<_{\mathrm{T}}$ B signifie que l'intervalle A est plus court que l'intervalle B. Par exemple « Janvier $<_{\mathrm{T}}$ Hiver » signifie que la durée du mois de Janvier est inférieure à la durée de l'hiver. Le prédicat « $=_{\mathrm{T}}$ » signifie que les intervalles qu'il compare ont approximativement la même durée, cela permet une certaine flexibilité pour comparer des intervalles aux durées similaires. On peut ainsi comparer Janvier et Février par ce prédicat bien qu'ils n'aient pas exactement la même durée.

Il est important de préciser que les prédicats temporels de comparaison, mis à part ceux exprimant des relations de durées ($<_{\mathrm{T}}$,$>_{\mathrm{T}}$ et $=_{\mathrm{T}}$), comparent deux intervalles périodiques ayant la même période. En effet, les relations d'ordre n'auraient plus de sens dans le cas contraire.

Les assertions temporelles

Les intervalles périodiques et les prédicats temporels de comparaison définis au dessus sont utilisés pour représenter des connaissances temporelles dans des assertions temporelles. Une assertion temporelle est une assertion qui représente une relation entre deux intervalles périodiques. C'est un donc un triplet composé de deux intervalles périodiques et d'un prédicat temporel de comparaison.

Définition 19 (Assertion temporelle)**.** *$\mathcal{P}_{\mathrm{t}}$ est l'ensemble des 21 prédicats temporels de comparaison. Une* assertion temporelle *est une assertion sous forme d'un triplet (e_1, p, e_2) tel que $p \in \mathcal{P}_{\mathrm{t}}$ et e_1 et e_2 sont des intervalles périodiques.*

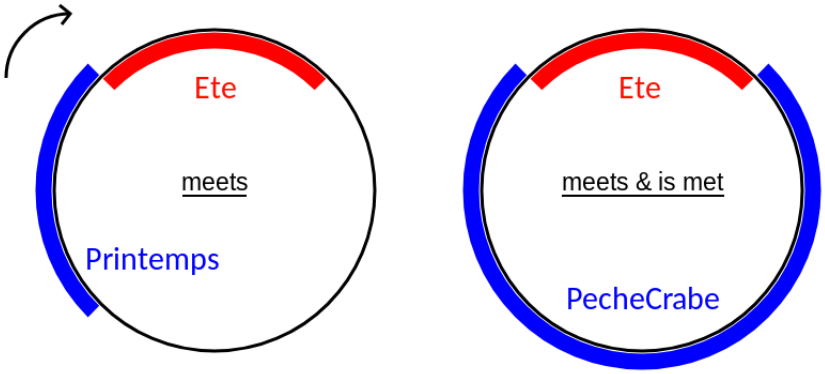


FIGURE 2.4 – Deux représentations cycliques d'assertion temporelles.

Exemple 21. *Comme nous l'avons vu précédemment, les intervalles périodiques sont souvent représentés sur un cercle qui se lit dans le sens des aiguilles d'une montre, les relations entre intervalles périodiques sont illustrées de manière similaire (figure 2.4). Le premier cercle représente l'assertion temporelle (Printemps, meets, Eté) et elle correspond à l'ordre ('Début-Printemps', 'FinPrintemps' = 'DébutEté', 'FinEté') de la deuxième ligne du tableau. Sa relation inverse est mi (is met by), ce qui donne (Eté, is met by, Printemps). Le second cercle illustre l'assertion temporelle (SaisonCrabe, meets&ismet, Eté).* `SaisonCrabe` *est relié à* `Eté` *par la relation 'meets&ismet', ce qui signifie que la saison du crabe commence quand l'été se termine et se termine quand l'été commence. Certains prédicats sont utilisés pour comparer des durées, par exemple dans l'assertion temporelle (Jour, $<_{\mathrm{T}}$, Mois) le prédicat de comparaison « $<_{\mathrm{T}}$ » est utilisé pour comparer la durée de* `Jour` *à celle de* `Mois`*.*

L'ontologie temporelle

Afin de représenter différentes connaissances temporelles, nous utilisons une ontologie. Une ontologie est une analogie d'une branche de la philosophie qui vise à étudier ce qui est

(« ὄντος » signifiant « étant » en grec), elles sont très utilisées pour décrire un domaine particulier. Les définitions d'ontologie en informatique peuvent varier mais conservent une structure similaire : Elles ont des connaissances générales et des connaissances assertionnelles, respectivement Tbox et Abox si l'on prend l'exemple des logiques de descriptions. Les connaissances générales sont composées de classes et de propriétés, les connaissances assertionnelles sont composées d'instances de classes et d'assertions de propriétés. Nous définissons ici une ontologie comme étant l'ensemble de ces quatre types de composants : les entités et les prédicats de comparaison sont les classes et les propriétés, les instances d'entités et les assertions sont les instances de classe et les assertions de propriétés.

Définition 20 (Ontologie)**.** *Une* ontologie *est un quadruplet* $\mathcal{O} = (\mathcal{C}, \mathcal{P}, \mathcal{E}, \mathcal{A})$ *tel que :*

- $\mathcal{C}$ *est l'ensemble des entités.*
- $\mathcal{P}$ *est l'ensemble des prédicats de comparaison.*
- $\mathcal{E}$ *est l'ensemble des instances.*
- $\mathcal{A}$ *est l'ensemble des assertions.*

Il existe beaucoup d'ontologies temporelles [Erm+14 ; LG89 ; Vie04 ; ZF02], comme par exemple l'ontologie temporelle OWL-Time [PH04], anciennement DAML-time [Hob+02], qui est maintenant une référence W3C [HP06] et également une des ontologies temporelles les plus utilisées. Il s'avère que ces ontologies temporelles ne prennent en compte ni les intervalles périodiques, ni bien sûr les relations qualitatives pour les comparer. C'est pourquoi, nous proposons ici une nouvelle ontologie temporelle qui prend en compte les intervalles périodiques et pourrait être intégrée à des ontologies temporelles existantes et plus conséquentes comme OWL-Time. Un article [Pa14] propose d'ailleurs d'intégrer les intervalles périodiques dans OWL-Time, il propose de définir la durée des sous-intervalles et des périodes mais ne propose pas de prendre en compte des relations qualitatives. Notre ontologie temporelle est composée de l'entité PeriodicInterval, des 21 prédicats temporels de comparaison, d'un ensemble d'instances de PeriodicInterval et d'un ensemble d'assertions temporelles sur ces instances.

Définition 21 (Ontologie Temporelle)**.** *Une* ontologie temporelle $\mathcal{O}_{\mathrm{t}} = (\mathcal{C}_{\mathrm{t}}, \mathcal{P}_{\mathrm{t}}, \mathcal{E}_{\mathrm{t}}, \mathcal{A}_{\mathrm{t}})$ *est une ontologie telle que :*

- $\mathcal{C}_{\mathrm{t}}$*={PeriodicInterval} est l'ensemble des entités temporelles.*
- $\mathcal{P}_{\mathrm{t}}$ *est l'ensemble des prédicats temporels de comparaison.*
- $\mathcal{E}_{\mathrm{t}}$ *est l'ensemble des intervalles périodiques.*
- $\mathcal{A}_{\mathrm{t}}$ *est l'ensemble des assertions temporelles.*

Exemple 22. *La figure 2.5 représente l'ontologie temporelle* $\mathcal{O}_{\mathrm{t}1}$ *sous forme de graphe orienté. Les intervalles périodiques de cette ontologie sont les étiquettes des sommets en jaune :* $\mathcal{E}_{\mathrm{t}1}$ *= {Printemps, SaisonCrabe, Annee...}. Les arcs représentent des assertions temporelles, ils*

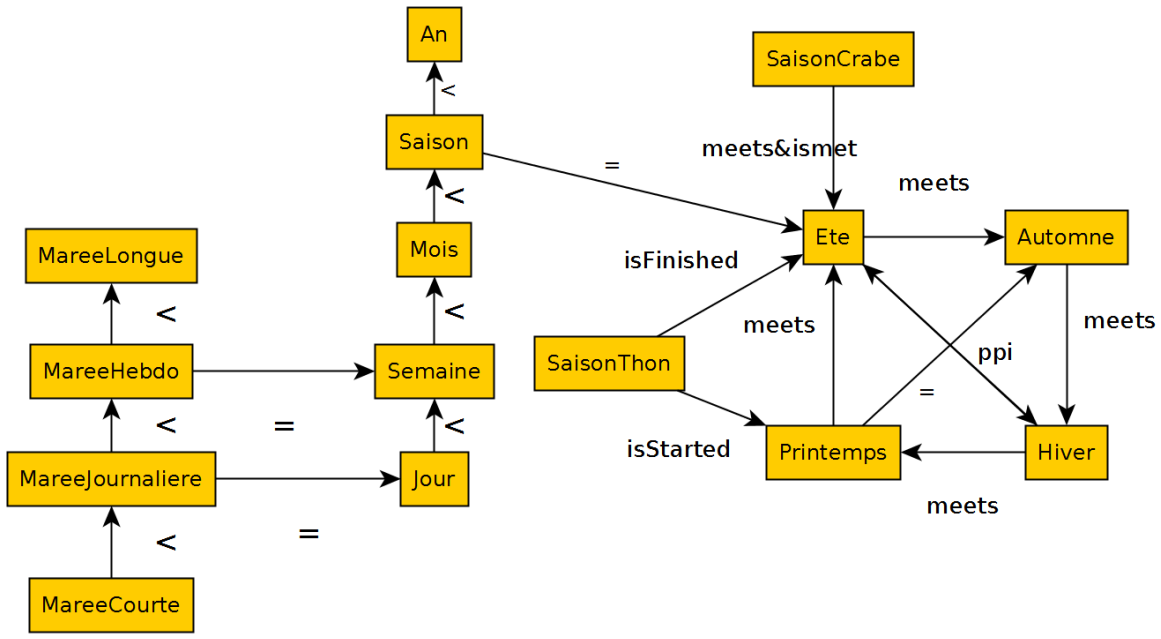


FIGURE 2.5 – Une représentation partielle de l'ontologie temporelle $\mathcal{O}_{t1}$.

sont étiquetés par les prédicats des assertions temporelles comme par exemple $<_T$ qui est le prédicat de l'assertion temporelle (MareeCourte[1]*, $<_T$, MareeJournaliere). La figure 2.5 est une représentation partielle de l'ontologie, car tous les prédicats n'y apparaissent pas forcément, mais aussi car certaines connaissances sont omises pour une visualisation plus claire. C'est pour cette raison qu'est par exemple représentée une assertion temporelle indiquant qu'une saison et l'été ont la même durée mais pas indiquant qu'une saison a la même durée que le printemps.*

2.2 La composante spatiale

Historiquement, en représentation des connaissances, les travaux sur le spatial ont connu plus de difficultés que les travaux sur le temps [Gal09]. Cela est dû à plusieurs raisons, notamment la dimension de l'espace qui est plus importante que celle du temps et par conséquent rend toute recherche plus complexe. Cela peut être illustré par la 'poverty conjecture' de Kenneth Forbus [For95 ; FNF90] qui stipule qu'il n'existe pas de représentation qualitative de l'espace qui ne soit pas dépendante d'un problème particulier. Une deuxième raison provient du fait que la flèche du temps a intrinsèquement un sens, contrairement à l'espace qui n'en a pas, ce qui complexifie encore le domaine spatial. Malgré cette complexité à représenter le domaine spatial, on peut tout de même faire un parallèle entre les structures temporelles et

1. Le mot « marée », dans le jargon de la pêche, correspond à une session en mer, d'où les durées journalières ou hebdomadaires des marées.

les structures spatiales. Effectivement, un instant ressemble à une localisation, un intervalle de temps à une région, une durée à une surface et caetera... Ces similitudes structurelles ont permis d'aider les travaux sur le domaine spatial qui ont pu se baser sur ceux effectués sur le domaine temporel [Gal09].

Alors que dans ce chapitre la composante temporelle considère un type spécifique d'entité temporelle, la composante spatiale considère un type plus abstrait d'entité spatiale, d'ailleurs nommé « entité spatiale ». Une entité spatiale désigne toute entité, physique ou non, ayant un caractère spatial, c'est une forme ayant une localisation dans l'espace. Ces entités spatiales sont définies comme les entités temporelles, en étant comparées en elles grâce à des prédicats spatiaux de comparaison. Des assertions spatiales peuvent ensuite être constituées pour décrire des entités spatiales dans une ontologie spatiale.

Cette section présente les entités spatiales, puis propose des prédicats spatiaux de comparaison permettant de former des assertions spatiales qui décrivent des relations entre les entités spatiales. Une ontologie spatiale composée d'assertions spatiales est finalement définie.

Les entités spatiales

Dans la plupart des ontologies spatiales, les entités spatiales sont abstraites et peu nombreuses. Les entités utilisées pour la représentation spatiale dans cette thèse sont des entités spatiales, celles-ci restent abstraites pour permettre de ne pas trop complexifier l'ontologie qui a pour but d'être associée au modèle des cartes cognitives dont la plus grande force est la simplicité. Dans le cas des entités temporelles, le choix d'utiliser une entité spécifique a été fait car les entités classiquement utilisées comme les instants, événements ou les intervalles classiques ne permettaient pas de répondre aux besoins des cartes cognitives, du moins pas aussi bien que le type spécifique des intervalles périodiques le permettait. Une entité spatiale est donc une entité abstraite, étant une forme dans un espace à trois dimensions, permettant de caractériser spatialement une autre entité qui a des propriétés spatiales.

Définition 22 (Entité spatiale)**.** *Une* entité spatiale *est une forme dans un espace à trois dimensions permettant de caractériser spatialement toute entité.*

Exemple 23. *Une entité spatiale est une forme, comme la forme d'une route qui est délimitée par des bords, qui a une surface, qui se situe à un endroit particulier dans un espace à 3 dimensions. Une entité spatiale est une forme qui peut être un point, comme une localisation particulière dans un espace à trois dimensions pouvant être désignée par des coordonnées GPS ou par une distance à une autre entité spatiale.*

Les prédicats spatiaux de comparaison

Les entités spatiales étant abstraites, le choix des prédicats de comparaison est d'autant plus important qu'il est seul moyen de décrire ces entités. Tout comme les prédicats temporels de comparaison qui sont de différentes natures (par exemple les relations topologiques et les relations de durées), les prédicats spatiaux de comparaison sont aussi de différentes natures : certains permettent d'exprimer des relations topologiques, d'autres de spécifier les distances entre entités ou de comparer leurs surfaces, d'autres prédicats représentent l'orientation des entités les unes par rapport aux autres. Les relations topologiques entre entités spatiales sont similaires aux relations d'Allen ou d'Osmani pour les intervalles périodiques. Il en existe deux familles très similaires (voir figure 2.6) apparues presque simultanément : les relations 9-intersections [Ege89 ; Ege91 ; EF91] et les relations RCC (Region Connection Calculus) [Coh+97 ; RC89 ; RCC92]. La première provient de la communauté scientifique d'information géographique et se base sur la géométrie élémentaire, tandis que la seconde, qui est basée sur la logique et se rapproche plus des formalismes utilisés pour les intervalles temporels, est plus simple [Hog+10] et fait maintenant office de référence dans le domaine de la représentation de connaissances[Gal09 ; GB07 ; Hog+10]. C'est pour cela qu'il est choisi dans cette thèse, d'utiliser les relations RCC. Les relations RCC-5, sont moins expressives que les relations RCC-8 du fait qu'elle regroupent certaines relations entre elles, par exemple DC(disconnected) et EC(externally connected) qui sont regroupées par la relation DR (Disjoint from) comme l'illustre la figure 2.6. Les relations RCC-8 ont été préférées aux relations RCC-5 principalement car elles offrent une plus grande expressivité. Bien qu'elles soient par conséquent plus complexes, l'utilisateur sera libre de se limiter à l'usage d'uniquement certaines d'entre elles.

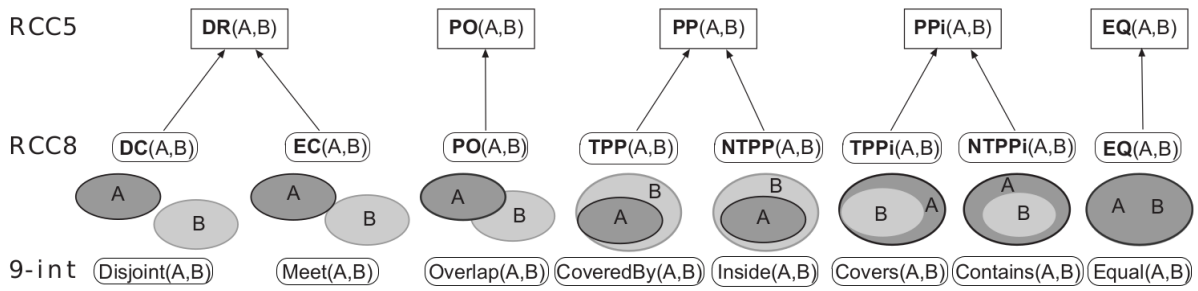


FIGURE 2.6 – Comparaison des modèles RCC-5, RCC-8 et 9-intersection [DBS05]

En plus de ces relations, quatre relations qualitatives de distances [Fra92] du type « est loin de » sont ajoutées, ainsi que trois relations similaires aux durées pour la partie temporelle qui servent à comparer des aires (comme par exemple celles des zones de pêche) et huit relations basées sur les points cardinaux [Fra92 ; Fra96] pour situer les entités les unes par rapport aux autres. En plus de ces relations, sont également considérées trois relations qualitatives de hauteur, profondeur, qui ont notamment un intérêt dans le domaine de la pêche.

De manière générale, un prédicat spatial de comparaison est une relation binaire sur des entités spatiales.

Définition 23 (Prédicat spatial de comparaison)**.** *Un* prédicat spatial de comparaison *est une relation binaire entre deux entités spatiales.* $\mathcal{P}_{\mathrm{s}}$ *est l'ensemble des 26 prédicats spatiaux de comparaison :*

$$\left\{ \begin{array}{c} DC, EQ, EC, PO, TPP, TPPi, NTPP, NTPPI, \\ very\ close,\ close,\ far,\ very\ far, \\ N, S, E, W, NE, NW, SE, SW, \\ <_{\mathrm{S}}, =_{\mathrm{S}}, >_{\mathrm{S}}, \\ il, hh, ih \end{array} \right\}$$

Les huit relations RCC-8, illustrées par la figure 2.7, considèrent une primitive de base $\mathcal{C}(x, y)$ nommée « connects » qui signifie que x et y partagent au moins un point. Toutes les relations sont construites à partir de cette primitive et des deux axiomes : $\mathcal{C}$ est réflexive et $\mathcal{C}$ est symétrique.

Bien que ces relations semblent complexes, elles sont particulièrement utiles pour décrire les relations topologiques entre différentes zones, comme par exemple pour exprimer qu'un parc éolien empiète sur une zone de pêche.

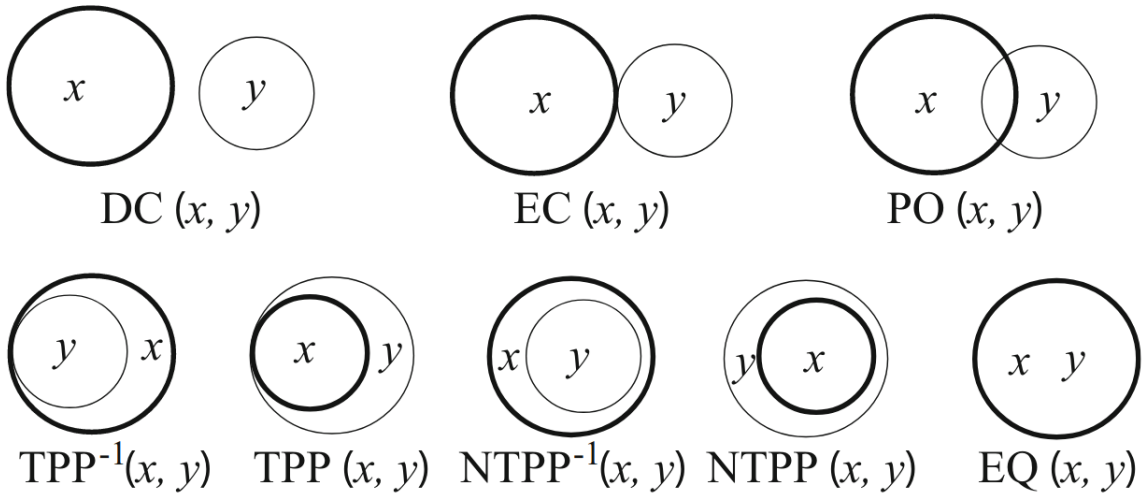


FIGURE 2.7 – Représentation des huit relations RCC-8 [Don08]

Les relations de distance permettent de donner un ordre de grandeur quant à la distance séparant deux entités. Il y a quatre relations de distance ordonnées de la plus petite à la plus grande :

very close $<$ *close* $<$ *far* $<$ *very far*.

Les relations cardinales sont souvent utilisées en représentation spatiale, elles permettent de situer les entités les unes par rapport aux autres en décrivant la direction de la seconde vers la première. Ainsi « A N B » signifie « A est au nord de B ».

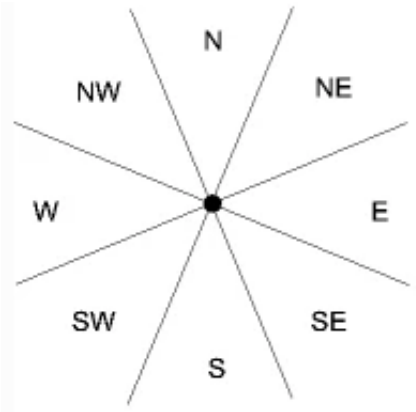


FIGURE 2.8 – Représentation canonique des relations cardinales [Gal09].

Trois relations permettent de comparer simplement des entités spatiales par rapport à la taille de leur surface :

— « x $<_{\mathrm{S}}$ y » signifie que x a une plus petite surface que y.

— « x $=_{\mathrm{S}}$ y » signifie que x et y ont la même surface, ou des surface similaires.

— « x $>_{\mathrm{S}}$ y » est la relation inverse de « x $<_{\mathrm{S}}$ y ».

Enfin, trois relations sont ajoutées pour permettre de comparer une position spatiale sur l'axe profondeur / altitude. Cela peut servir par exemple à comparer des profondeurs de zones de pêche, ou des différences d'altitudes entre différents types de satellites :

— « x *il (is lower)* y » signifie x est plus bas, ou moins haut, que y.

— « x *hh (has height)* y » signifie que x et y ont la même altitude ou la même profondeur.

— « x *ih (is higher)* y » est la relation inverse de « x *il* y ».

La grande diversité des prédicats spatiaux de comparaisons provient majoritairement de la complexité du domaine spatial, qui rend impossible une représentation exhaustive grâce à un seul type de relations. La plupart restent néanmoins très simples comme ceux permettant de comparer des surfaces ou distances.

Les assertions spatiales

Les entités spatiales et les prédicats spatiaux de comparaison définis précédemment sont utilisés pour représenter des connaissances spatiales dans des assertions spatiales. Une assertion spatiale est une assertion qui représente une relation entre deux entités spatiales grâce

à un prédicat spatial de comparaison. Cest un donc un triplet composé de deux entités spatiales et dun prédicat spatial de comparaison.

Définition 24 (Assertion spatiale)**.** $\mathcal{P}_{\mathrm{s}}$ *est l'ensemble des 26 prédicats spatiaux de comparaison. Une* assertion spatiale *est une assertion qui constitue un triplet* (e_1, p, e_2) *tel que* $p \in \mathcal{P}_{\mathrm{s}}$ *et* e_1 *et* e_2 *sont des entités spatiales.*

Exemple 24. *(*PetiteZonePeche*,* $<_{\mathrm{S}}$*,* GrandeZonePeche*) est une assertion spatiale. Elle utilise le prédicat spatial de comparaison des tailles de surfaces* $<_{\mathrm{S}}$ *pour comparer les entités spatiales* PetiteZonePeche *et* GrandeZonePeche*. Sa signification est trivialement qu'une petite zone de pêche a une plus petite surface qu'une grande zone de pêche.*

*(*StNazaire*,* E*,* LeCroisic*) est une autre assertion spatiale qui utilise le prédicat spatial de comparaison de points cardinaux* E (East) *pour comparer les entités spatiales* StNazaire *et* LeCroisic*. Sa signification est que St Nazaire se situe à l'est du Croisic.*

L'ontologie spatiale

Il existe beaucoup d'ontologies spatiales mais il n'existe cependant pas une ontologie faisant office de référence. Le W3C a néanmoins publié un document [LSG07] décrivant les standards ISO [2] et OGC [3] à ce sujet et un autre document [Van+19] donnant les avantages, inconvénients et exigences attendues d'une ontologie spatiale de référence. Ces exigences attendues sont très générales, comme le fait que l'ontologie doit être simple, évaluée par les personnes de domaines concernés, et qu'elle doit être aussi une référence OGC. Parmi les ontologies spatiales existantes on peut en citer trois faisant partie des plus connues. geoSPARQL [BK11] est une extension géographique du langage SPARQL dont fait partie une ontologie contenant deux classes « Feature » et « Geometry ». L'ISA Programme Location Core Vocabulary [PLA13] contient les trois classes « Adress », « Location » et « Geometry », elle concernent principalement les descriptions d'adresses. NeoGeo [Sal+11] est une autre ontologie spatiale, de petite taille, elle contient une seule classe et des relations topologiques du type RCC8 ou 9-intersection. Il ne semble pas exister d'ontologie spatiale utilisant tous les prédicats de comparaison présentés dans cette section, mais certaines en utilisent quelques-uns, notamment NeoGeo et geoSPARQL qui utilisent les relations topologiques.

Cette section définit une ontologie spatiale ayant pour seule classe : « SpatialEntity » et les 26 prédicats spatiaux de comparaison définis précédemment. De manière analogue à l'ontologie temporelle, l'ontologie spatiale contient un ensemble d'instances de « SpatialEntity » et un ensemble d'assertions spatiales les décrivant.

Définition 25 (Ontologie Spatiale)**.** *Une* ontologie spatiale $\mathcal{O}_{\mathrm{s}} = (\mathcal{C}_{\mathrm{s}}, \mathcal{P}_{\mathrm{s}}, \mathcal{E}_{\mathrm{s}}, \mathcal{A}_{\mathrm{s}})$ *est une ontologie telle que :*

2. International Organization for Standardization
3. Open Geospatial Consortium

— $\mathcal{C}_s$={SpatialEntity} *est l'ensemble des entités spatiales.*

— $\mathcal{P}_s$ *est l'ensemble des prédicats spatiaux de comparaison.*

— $\mathcal{E}_s$ *est l'ensemble des instances de l'ontologie.*

— $\mathcal{A}_s$ *est l'ensemble des assertions spatiales.*

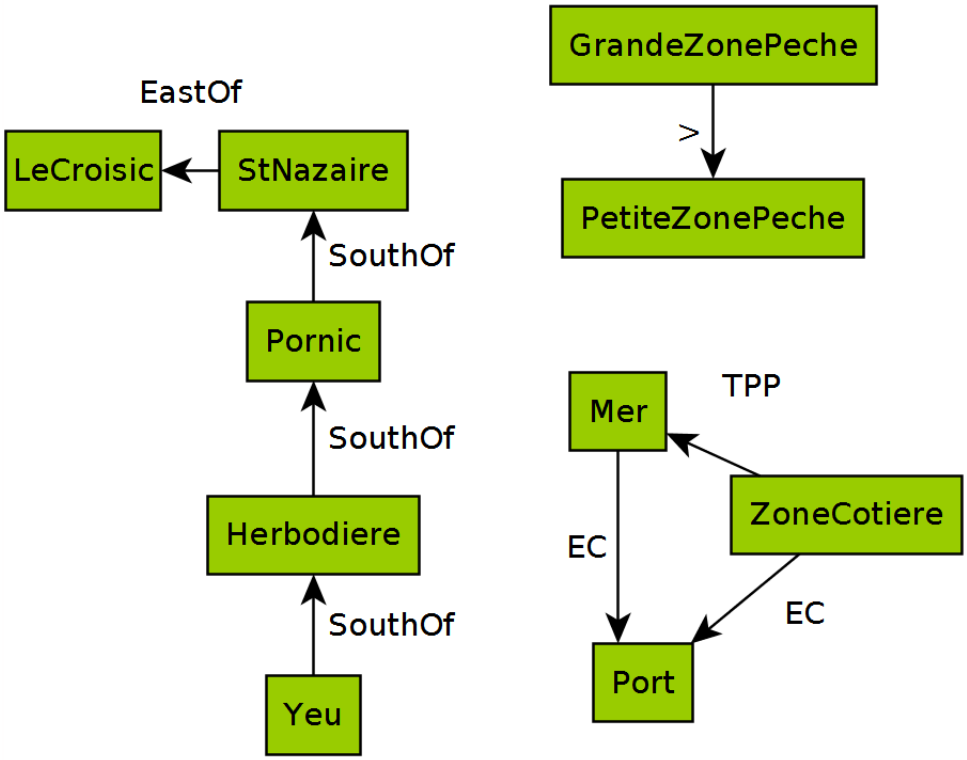


FIGURE 2.9 – Une représentation partielle de l'ontologie spatiale $\mathcal{O}_{s1}$.

Exemple 25. *La figure 2.9 représente l'ontologie spatiale $\mathcal{O}_{s1}$ sous forme de graphe orienté. Les entités spatiales de cette ontologie sont les étiquettes des sommets en vert : $\mathcal{E}_s$= {Yeu, LeCroisic, PetiteZonePeche, Port...}. Les arcs représentent des assertions spatiales, ils sont étiquetés par les prédicats des assertions spatiales comme par exemple* southOf *et* EC *qui sont les prédicats des assertions spatiales (Yeu,southOf,Herbodière) et (Mer, EC, Port).*

2.3 Les cartes cognitives ontologiques : des cartes cognitives à composantes taxonomique, temporelle et spatiale

Lors du précédent chapitre, nous avons vu qu'une carte cognitive taxonomique associe une taxonomie de concepts à une carte cognitive dans le but d'organiser les concepts entre eux et

de les caractériser par des relations de subsomption. En plus de favoriser la bonne construction de plusieurs cartes, cela a permis de formaliser de nouvelles opérations comme la vue et la synthèse de cartes cognitives. De la même manière, cette partie vise à utiliser les ontologies spatiales et temporelles pour caractériser temporellement et spatialement les concepts des cartes cognitives. Cette partie regroupe dans un premier temps les deux ontologies en une ontologie spatio-temporelle avant de définir une carte cognitive ontologique comme une carte cognitive taxonomique définie sur cette ontologie spatio-temporelle.

L'ontologie spatio-temporelle

Les parties 1 et 2 de ce chapitre présentent respectivement les ontologies temporelle et spatiale. Bien qu'elles ne décrivent pas les mêmes choses, ces deux ontologies sont très similaires. Dans un but de clarté, nous définissons ici une ontologie spatio-temporelle qui regroupe ces deux ontologies. Ainsi, une ontologie spatio-temporelle est une ontologie ayant deux classes : PeriodicInterval et SpatialFeature. L'ensemble de ses prédicats de comparaison est l'union de l'ensemble des prédicats temporels et spatiaux. Ses instances sont des intervalles périodiques et des entités spatiales. Celles-ci sont décrites par l'ensemble des assertions spatio-temporelles qui est l'union de l'ensemble des assertions temporelles et de l'ensemble des assertions spatiales.

Définition 26 (Ontologie spatio-temporelle)**.** *Soient* $\mathcal{O}_{\mathrm{t}} = (\mathcal{C}_{\mathrm{t}}, \mathcal{P}_{\mathrm{t}}, \mathcal{E}_{\mathrm{t}}, \mathcal{A}_{\mathrm{t}})$ *une ontologie temporelle et* $\mathcal{O}_{\mathrm{s}} = (\mathcal{C}_{\mathrm{s}}, \mathcal{P}_{\mathrm{s}}, \mathcal{E}_{\mathrm{s}}, \mathcal{A}_{\mathrm{s}})$ *une ontologie spatiale. Une* ontologie spatio-temporelle $\mathcal{O}_{\mathrm{st}} = (\mathcal{C}_{\mathrm{st}}, \mathcal{P}_{\mathrm{st}}, \mathcal{E}_{\mathrm{st}}, \mathcal{A}_{\mathrm{st}})$ *est une ontologie combinant une ontologie temporelle et une ontologie spatiale telle que :*

- $\mathcal{C}_{\mathrm{st}} = \mathcal{C}_{\mathrm{t}} \cup \mathcal{C}_{\mathrm{s}}$*= {PeriodicInterval, SpatialFeature}*
  est l'ensemble des entités spatio-temporelles.
- $\mathcal{P}_{\mathrm{st}} = \mathcal{P}_{\mathrm{t}} \cup \mathcal{P}_{\mathrm{s}}$ *est l'ensemble des prédicats spatio-temporels de comparaison.*
- $\mathcal{E}_{\mathrm{st}} = \mathcal{E}_{\mathrm{t}} \cup \mathcal{P}_{\mathrm{s}}$ *est l'ensemble des instances d'entités spatio-temporelles.*
- $\mathcal{A}_{\mathrm{st}} = \mathcal{A}_{\mathrm{t}} \cup \mathcal{A}_{\mathrm{s}}$ *est l'ensemble des assertions spatio-temporelles.*

Exemple 26. *Considérons* $\mathcal{O}_{\mathrm{st1}}$ *comme étant l'ontologie spatio-temporelle résultant de l'union de l'ontologie temporelle* $\mathcal{O}_{\mathrm{t1}}$ *(figure 2.5) et de l'ontologie spatiale* $\mathcal{O}_{\mathrm{s1}}$ *(figure 2.9). Cette ontologie contient des assertions spatio-temporelles comme (*Printemps*,* meets*,* Ete*) ou (*StNazaire*,* southOf*,* LeCroisic*).*

Notez que, d'un point de vue de l'utilisation du modèle, certaines entités et assertions sont génériques et présentes par défaut comme les saisons et leur relations tandis que d'autres sont spécifiques à un domaine et au cas d'utilisation comme la saison de la pêche au crabe ou les tailles de zones de pêche.

Carte cognitive ontologique

Une carte cognitive ontologique est une carte cognitive taxonomique définie sur une ontologie spatio-temporelle. En plus d'un concept, chaque noeud de la carte est étiqueté par deux entités : un intervalle périodique et une entité spatiale. Ces deux entités sont éventuellement liées à l'ontologie spatio-temporelle grâce à un ensemble d'assertions spatio-temporelles appartenant à la carte. De cette manière, les noeuds de la carte cognitive peuvent être caractérisés temporellement et spatialement si besoin.

Définition 27 (Carte Cognitive Ontologique)**.** *Soit $\mathcal{O}_{\mathrm{st}} = (\mathcal{C}_{\mathrm{st}}, \mathcal{P}_{\mathrm{st}}, \mathcal{E}_{\mathrm{st}}, \mathcal{A}_{\mathrm{st}})$ une ontologie spatio-temporelle. Soit $CM = (N, A, labelN, labelE)$ une carte cognitive taxonomique définie sur une taxonomie $T = (C, \leqslant)$ et un ensemble de valeurs I. Une* carte cognitive ontologique OCM *définie sur $\mathcal{O}_{\mathrm{st}}$ est un quadruplet $(CM, labelT, labelS, \mathcal{A}_{OCM})$ tel que :*

- *$labelT$: est une fonction d'étiquetage de la carte qui attribue à un noeud $n \in N$ un unique intervalle périodique et_n.*
- *$labelS$: est une fonction d'étiquetage de la carte qui attribue à un noeud $n \in N$ une unique entité spatiale es_n.*
- *$\mathcal{A}_{OCM}$ est un ensemble d'assertions spatio-temporelles (e_1, p, e_2) où : $\exists n \in N, (labelT(n) = e_1 \vee\ labelS(n) = e_1)$ et $e_2 \in \mathcal{E}_{\mathrm{st}}$.*

Une carte cognitive ontologique contient des connaissances variées, ce qui peut compliquer sa visualisation. Un premier mode de visualisation est proposé, il représente l'intégralité du modèle et permet de comprendre la structure et le positionnement des assertions spatio-temporelles. Il est cependant très peu lisible, toutes les connaissances étant représentées. Un second mode de visualisation est ensuite proposé, il ne représente plus que la carte cognitive avec les assertions des cartes positionnées sous les noeuds, celui-ci illustre moins bien la structure du modèle mais offre plus de clarté pour visualiser le contenu d'une carte cognitive ontologique.

Exemple 27. *Cet exemple décrit la figure 2.10 qui représente la carte cognitive ontologique OCM1. La carte cognitive est en dessous de la taxonomie de concepts et à gauche de l'ontologie spatio-temporelle sur laquelle elle est définie. Chaque sommet de la carte est étiqueté par deux entités : un intervalle périodique et une entité spatiale. Elles sont toutes les deux représentées en dessous du noeud par des cercles, respectivement de couleur jaune et verte. Les assertions spatio-temporelles de la carte cognitive ontologique sont des triplets rattachés à la carte. Conformément à la définition, le premier élément du triplet est une entité associée à un noeud de la carte, le deuxième un prédicat de comparaison et le troisième une entité de l'ontologie. Les assertions spatio-temporelles sont ici représentées par les arcs orientés de couleur rouge allant de la carte cognitive vers l'ontologie. De manière plus précise, ces arcs relient soit un intervalle périodique associé à un noeud de la carte et un intervalle périodique de l'ontologie spatio-temporelle, soit une entité spatiale associée à un noeud de la carte et une autre de*

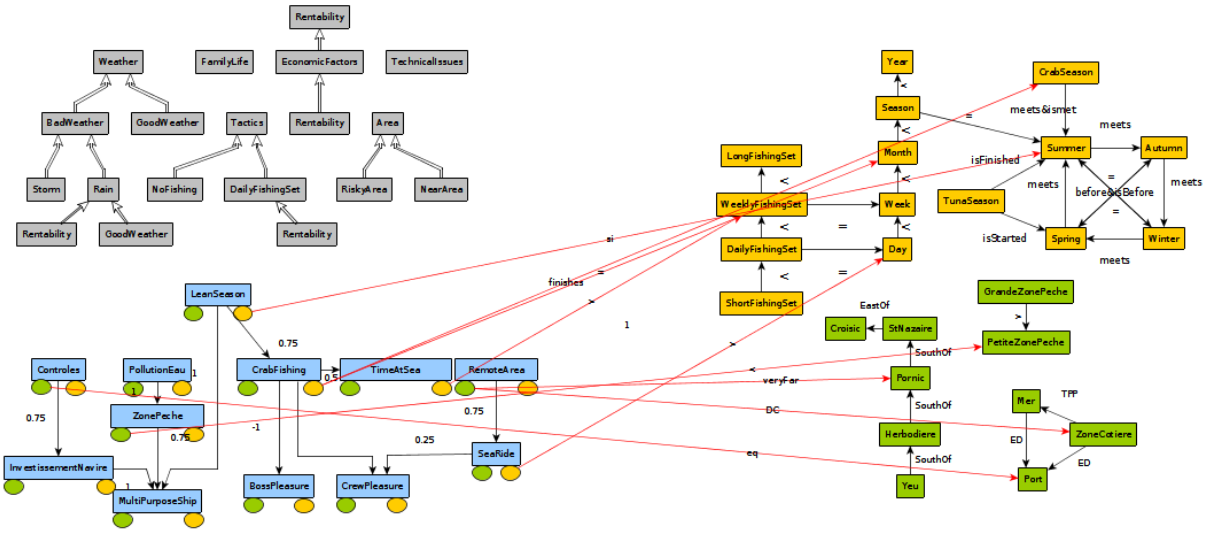


FIGURE 2.10 – Carte cognitive ontologique OCM1.

l'ontologie spatio-temporelle. Les entités associées aux noeuds ne sont donc pas directement comparables entre-elles : toutes les assertions spatio-temporelles de la carte (c'est-à-dire les arcs orientés rouges) vont de la carte vers l'ontologie. Notez que certaines entités associées aux noeuds ne sont pas décrites par une assertion spatio-temporelle mais sont tout de même présentes, chaque noeud est en effet étiqueté par une entité temporelle et une entité spatiale, qu'il soit caractérisé ou non.

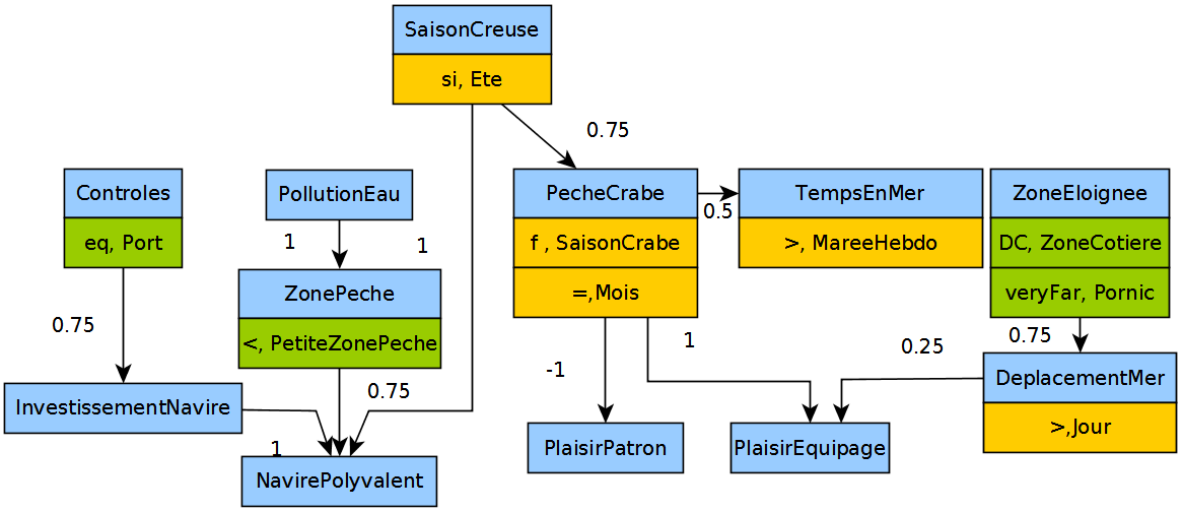


FIGURE 2.11 – Représentation graphique de la carte cognitive spatio-temporelle OCM1.

Exemple 28. *La représentation graphique de la figure 2.10 ne permet pas de bien visualiser les assertions spatio-temporelles. La figure 2.11 illustre une autre représentation graphique*

d'une carte cognitive ontologique en représentant la même carte OCM1. Dans ce mode de visualisation, les assertions des cartes et de l'ontologie sont représentées différemment. Une assertion spatio-temporelle d'une carte cognitive est représentée visuellement par un rectangle sous le noeud qu'elle caractérise. Les entités associées aux noeuds sont omises, c'est pour cela que les assertions spatio-temporelles sont écrites comme des couples et non des triplets. Par exemple, le noeud étiqueté par `SaisonCreuse` *(en haut) est étiqueté par un intervalle périodique* et_1 *et caractérisé par l'assertion temporelle (*et_1*, si, Ete) où 'si' est le prédicat temporel de comparaison isStartedBy. Notez que plusieurs assertions temporelles ou spatiales peuvent être associées au même noeud, comme c'est le cas du noeud étiqueté par* `PecheCrabe`*. Ce noeud est caractérisé par un intervalle périodique qui dure un mois (*$=_{\mathrm{T}}$*,Mois) à la fin de la saison du crabe (f,SaisonCrabe). Ce pêcheur pêche donc le crabe un mois à la fin de la saison du crabe.*

Notez qu'il est tout à fait possible d'avoir dans la même carte deux noeuds étiquetés par le même concept. Un même concept qui apparaît plusieurs fois dans une carte peut donc avoir différentes influences en fonction de ses caractérisations temporelles et spatiales.

Sémantique des influences

L'information principale représentée par une carte cognitive est l'ensemble des influences, c'est le coeur du modèle, les influences ont donc une importance particulière. Lorsque qu'un concept est caractérisé spatialement ou temporellement et qu'il joue un rôle dans une influence d'une carte, cette influence prend une sémantique particulière. C'est ce sur quoi s'attarde cette partie, en décrivant le sens des influences se trouvant dans ce cas.

Dans notre modèle une influence n'est pas caractérisée spatialement ou temporellement, les concepts le sont. On parle d'une influence caractérisée quand au moins un de ses concepts est caractérisé spatio-temporellement. Une influence caractérisée A→B est polysémique, elle a en effet deux sens possibles, considérant la caractérisation spatio-temporelle comme étant soit une simple description des concepts, soit un renseignement particulier sur les objets de l'influence. Dans le premier cas, A→B signifie « le concept A influence le concept B » et les concepts A et B peuvent être décrits au travers d'assertions spatio-temporelles, cela ne change pas grandement le sens de l'influence si ce n'est que l'influence reste vraie seulement dans le contexte spatio-temporel de ses concepts. Dans le second, A→B signifie « le fait que le concept A soit selon ses caractérisations spatio-temporelles influence le fait que le concept B soit selon les siennes », auquel cas l'influence agirait sur les propriétés spatiales et temporelles de ces concepts.
Bien sûr, ces deux distinctions proviennent de l'interprétation des concepts caractérisés dans une influence, il y a donc quatre cas possibles d'interprétation pour une influence caractérisée.

Exemple 29. *Cet exemple s'appuie sur l'influence caractérisée de la figure 2.12. C'est une*

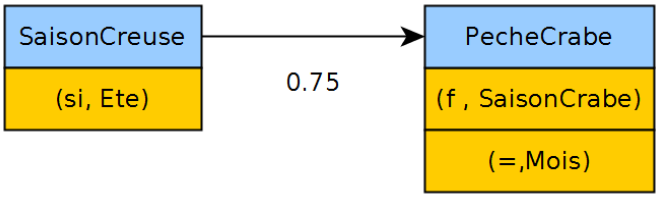


FIGURE 2.12 – Une influence caractérisée de la carte cognitive ontologie OCM1.

influence du concept $\mathtt{SaisonCreuse}$*, vers le concept* $\mathtt{PecheCrabe}$ *étiquetée par la valeur positive 0,75. Le concept* $\mathtt{SaisonCreuse}$ *est caractérisé temporellement par une assertion temporelle qui signifie que la saison creuse est démarrée (is started by) par l'été, c'est-à-dire qu'elle commence au début de l'été puis dure après la fin de l'été. Le concept* $\mathtt{PecheCrabe}$ *est également caractérisé temporellement par deux assertions temporelles qui signifient que la pèche au crabe dure un mois (*$=_{\mathrm{T}}$*,Mois) à la fin de la saison du crabe (finishes, SaisonCrabe). On peut interpréter cette influence comme étant simplement une influence entre les concepts* $\mathtt{SaisonCreuse}$ *et* $\mathtt{PecheCrabe}$ *qui sont accessoirement décrits par leur aspects temporels. On peut également interpréter cette influence par la signification suivante : « Le fait que la saison creuse soit pendant l'été et continue ensuite, influence le fait que la pêche au crabe dure un mois à la fin de la saison du crabe. ».*

Cette ambiguïté des influences caractérisées est simplement une forme de l'ambiguïté déjà présente dans une influence classique concernant ses objets, c'est-à-dire les propriétés des concepts qui sont impliquées dans l'influence. Les influences caractérisées réduisent cependant l'ambiguïté du contexte de l'influence, puisque les informations spatiales et temporelles n'ont plus à être implicites.

Conclusion

Dans ce chapitre, nous avons présenté une ontologie spatio-temporelle, composée des deux ontologies, temporelle et spatiale, ayant une structure commune. Cette ontologie permet de représenter différentes entités spatiales ou temporelles d'un domaine concerné qui peuvent être utilisées pour décrire les concepts d'une carte cognitive. Nous avons vu que la description de ces concepts permettait de donner un sens particulier aux influences. Ces nouvelles informations peuvent être représentées visuellement tout en conservant une bonne lisibilité. Cette modélisation est aussi très utile pour effectuer des taches de raisonnement, c'est ce sur quoi porte le chapitre suivant qui pose des bases de raisonnement en utilisant des primitives pour accéder aux informations du modèle, notamment les informations spatio-temporelles.

La bibliographie ne propose quasiment aucune solution pour prendre en compte les aspects temporels et spatiaux dans les cartes cognitives. Quelques articles traitent de l'intégration des

aspects temporels dans les cartes cognitives, comme [Hag92] et [PK95] qui concernent le délai d'une influence dans les cartes cognitives floues. D'autres articles proposent de spécifier un temps de base, c'est-à-dire un ordre de grandeur des durées des influences [CT01 ; Zho+08]. Un article [TM97], propose aussi de considérer une dégradation de l'influence au cours du temps. Tous ces articles concernent la temporalité des influences, que ce soit leur délai, leur dégradation ou leur temps de base, mais aucun ne concerne la temporalité des concepts. Concernant les aspects spatiaux, seuls quelques articles de recherche en géographie proposent de générer une carte cognitive floue contextuelle à partir d'un système SIG (système d'information géographique) [LS99 ; SL99], mais ils ne permettent pas de caractériser les concepts sur leurs aspects spatiaux. Caractériser les concepts offre plus d'expressivité que de caractériser les influences, en effet les caractérisations des concepts permettent aussi de spécifier la sémantique des influences entre ces concepts.

CHAPITRE 3

LES PRIMITIVES : DES ACCÈS AUX CONNAISSANCES DES CARTES COGNITIVES ONTOLOGIQUES

Introduction

Un modèle de représentation de connaissances doit fournir des outils permettant son exploitation par des utilisateurs ou d'autres systèmes. L'exploitation d'un modèle comprend d'abord la restitution de ses connaissances par un accès à celles-ci. L'accès aux connaissances ne doit pas être une restitution brute de toutes les connaissances contenues dans le modèle auquel cas ces connaissances ne seraient pas directement exploitables, principalement à cause de leur quantité. L'accès aux connaissances doit donc être précis et affiné pour une bonne efficacité. C'est autour de cette problématique que se positionne ce chapitre.

Le modèle des *cartes cognitives ontologiques*, que nous avons défini dans le chapitre précédent, représente des connaissances complexes et variées : 1) la *taxonomie* de concepts représente des concepts et des relations de spécialisation entre ces concepts, 2) l'*ontologie spatio-temporelle* représente des connaissances temporelles et spatiales à l'aide d'entités et de nombreux prédicats de comparaison, 3) la carte cognitive représente des influences entre concepts en intégrant les connaissances de la taxonomie et de l'ontologie spatio-temporelle. Toutes ces connaissances, en plus d'être complexes et variées, peuvent être nombreuses, notamment lorsque l'on considère un ensemble de cartes [Ede04].

La contribution présentée dans ce chapitre pose les bases pour fournir un accès direct aux connaissances du modèle. L'approche consiste à créer une représentation structurée du modèle lui-même pour permettre de manipuler les connaissances ou raisonner dessus. Cette représentation structurée est formée d'un ensemble de *primitives*. Les primitives sont des relations entre les objets du modèle des cartes cognitives ontologiques. Ces objets peuvent être des cartes, des concepts, des influences, des chemins d'influence, des valeurs d'influences propagées ou non, des entités temporelles ou spatiales, des prédicats ou des quantités. Les primitives permettent de décrire les liens entre ces différents objets, comme par exemple le lien d'appartenance d'un concept à une carte cognitive, ou celui d'un chemin d'influence à son

concept-destination ou à sa longueur. Certaines primitives donnent accès aux connaissances du modèle classique des cartes cognitives [RGL18a], d'autres concernent les connaissances taxonomiques [RGL18b], enfin d'autres primitives donnent accès aux connaissances spatiales et temporelles du modèle des cartes cognitives ontologiques [RGL20b]. Il est possible d'accéder aux connaissances désirées en les spécifiant au travers de *formules primitives*. Une formule primitive est une expression syntaxique, qui peut s'apparenter à une requête très basique. Elle permet de spécifier les différents attributs d'une primitive par des variables ou des constantes afin de contraindre la valeur des variables.

Ce chapitre présente dans une première partie les primitives d'accès aux connaissances du modèle classique des cartes cognitives. La deuxième partie présente les primitives d'accès aux connaissances relatives à la taxonomie de concepts. La troisième partie définit les primitives d'accès aux connaissances temporelles. Enfin, la dernière partie définit les primitives d'accès aux connaissances spatiales.

3.1 Primitives d'accès aux cartes cognitives

Cette première partie introduit l'idée de primitive et de formule primitive avant de proposer 8 primitives d'accès aux connaissances du modèle classique des cartes cognitives. Ces connaissances concernent les concepts, les influences, les chemins d'influence et les valeurs d'influence.

Bien que cette partie présente les primitives permettant l'accès aux connaissances du modèle classique des cartes cognitives, il est introduit ici un ensemble de cartes cognitives ontologiques car étant le modèle le plus complet, toutes les primitives sont définies dessus.

Notation 1 (Ensemble de cartes cognitives ontologiques)**.** *Un* ensemble de cartes cognitives ontologiques $\mathcal{S} = (OCM_1, ..., OCM_n)$ *fait référence à un ensemble de cartes cognitives ontologiques définies sur le même ensemble de valeurs I, la même taxonomie T et la même ontologie spatio-temporelle* $\mathcal{O}_{\mathrm{st}}$*.*

Exemple 30. *L'ensemble de cartes cognitives ontologiques utilisé dans ce chapitre est* $\mathcal{S}_1$*, il est composé des cartes cognitives ontologiques OCM1 (figure 2.11 du chapitre 2) et OCM2 (figure 3.1) définies sur l'ensemble de valeurs I=[-1 ;1], la taxonomie T1 (figure 3.2) et l'ontologie spatio-temporelle du chapitre précédent* $\mathcal{O}_{\mathrm{st}1}$ *(figures 2.5 et 2.9).*

Une primitive est une relation dont les attributs sont associés à des domaines qui sont les objets du modèle des cartes cognitives ontologiques. Ces objets sont les différents composants du modèle : les cartes, les concepts, les influences, les valeurs... Par exemple le domaine de l'attribut carte est l'ensemble de cartes cognitives ontologiques et le domaine de l'attribut

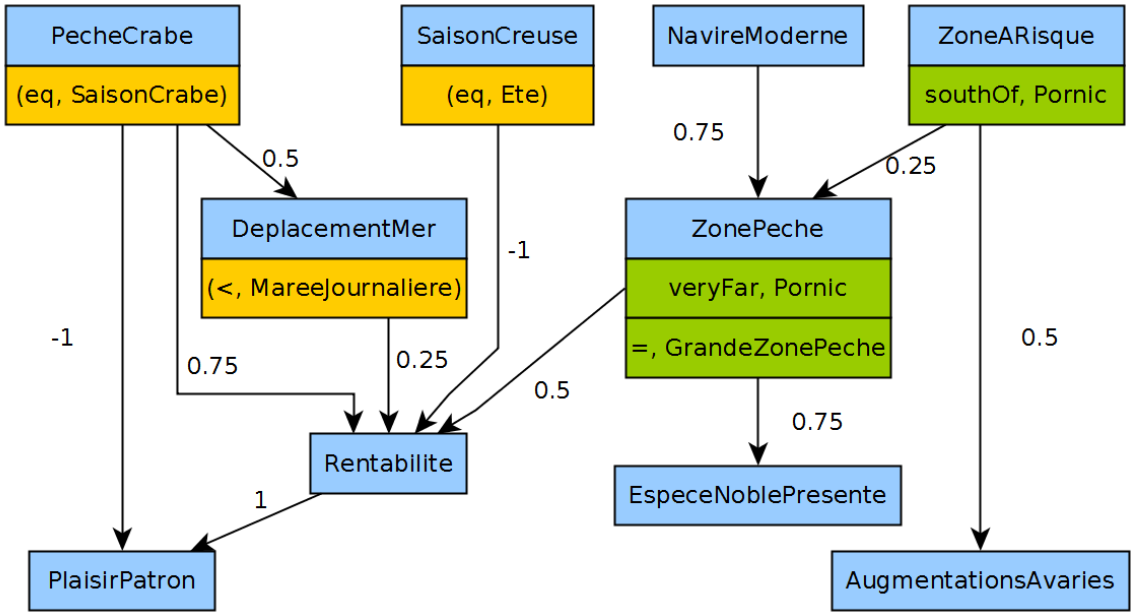


FIGURE 3.1 – Carte cognitive ontologique OCM2 de l'ensemble de cartes cognitives ontologiques $\mathcal{S}1$.

concept est l'ensemble des concepts de la taxonomie. La valeur d'une primitive est un sous-ensemble du produit cartésien des domaines des attributs qui la composent. Ce sous-ensemble est défini pour chaque primitive, c'est ce qui fait son sens. Une primitive est une relation, sa définition est basée sur celle du calcul relationnel de domaine [AD93 ; LP82] dans laquelle un tuple est défini comme une fonction des attributs vers les éléments de leur domaine

Définition 28 (Primitive)**.** *Soient* $A = \{A_1, A_2, ..., A_n\}$ *un ensemble d'attributs et* $D = \{D_1, D_2, ..., D_n\}$ *un ensemble de domaines.*
Une primitive *est une relation* $R(A_1 : D_1, ..., A_n : D_n)$*, d'arité n telle que sa valeur est un sous-ensemble du produit cartésien indexé par ses attributs :*

$$val(R) \subseteq \{t : A \to D_1 \cup ... \cup D_n \mid \forall i \in [1; n],\ t(A_i) \in D_i\}.$$

Une formule primitive est une expression syntaxique qui se base sur une primitive. Elle sert à exprimer des ensembles particuliers comprenant des objets du modèle des cartes cognitives ontologiques. Pour cela elle permet, en remplaçant les attributs des primitives soit par des variables soit par des constantes, de spécifier le sens de la formule primitive. Une formule primitive étant une expression syntaxique, c'est-à-dire un élément de langage, elle est définie par sa syntaxe et sa sémantique.

Une formule primitive est composée du nom d'une primitive suivi des attributs de la primitive séparés par des virgules et compris entre parenthèses. Chaque attribut est représenté soit par une constante, soit par une variable qui sera désignée par son nom et préfixée par convention

par un point d'interrogation de manière similaire à l'expression de variables en SPARQL [AG08 ; PAG09 ; PS06].

La sens d'une formule primitive est l'ensemble des tuples contenant les valeurs de ses variables.

Définition 29 (Formule primitive)**.** *Une formule primitive est une expression syntaxique désignant une primitive.* $T = \{T1, ..., Tn\}$ *est l'ensemble des termes, c'est-à-dire l'ensemble des variables* V *et des constantes désignant les attributs.*
Sa forme syntaxique est la suivante :

```
nomDePrimitive '(' ('?'varName | cst) [','('?'varName | cst)]* ')'
```

où nomDePrimitive, varName et cst sont respectivement un nom de primitive, un nom de variable et une constante.
La syntaxe abstraite suivante est utlisée : $P(A1 : T1, ..., An : Tn)$*, où* P *est le nom de la primitive et* A *et* T *sont les paires attribut-terme(variable ou constante).*
Son sens est le suivant[1] [2] *:*

$$mng_F(P(A1 : T1, ..., An : Tn)) =$$
$$\{\, t \in \prod(V : D) \mid \exists\, t1 \in val(P),\ \forall i \in [1; n],$$
$$t1(Ai) = Ti \ si \ Ti \ est \ une \ constante,$$
$$t1(Ai) = t(Ai) \ si \ Ti \ est \ une \ variable\}$$

Exemple 31. *Définissons à titre d'exemple la primitive* $Double(n1 : \mathbb{N}, n2 : \mathbb{N})$ *qui lie chaque entier naturel à son double. On peut constituer plusieurs formules primitives bien formées concernant cette primitive. Par exemple* `Double(?v1,10)` *dont le premier attribut est une variable étant donné qu'il est préfixé par un point d'interrogation et le second est la constante 10. Son sens est un ensemble de tuples. Chaque tuple est représenté par une fonction des attributs* A_1 *(n1) désignés par les variables (v1) vers les éléments des domaines de ces attributs (*$\mathbb{N}$*). Cet ensemble de tuples est ici un singleton : {t1} avec* $t1(A_1) = 5$*, noté plus simplement par l'ensemble contenant les images des ses éléments : {(5)}. La formule primitive* `Double(?n1,?n2)` *a ses deux attributs exprimés par des variables, son sens est l'ensemble de couples : {(0,0),(1,2),(2,4)...}.*

IsInMap est une primitive qui permet d'accéder aux concepts et aux cartes dans lesquels ils apparaissent. C'est une primitive qui relie les cartes cognitives aux concepts par lesquels sont étiquetés leurs noeuds.

Définition 30 (Primitive IsInMap)**.** *Soit* $\mathcal{S}$ *un ensemble de cartes cognitives ontologiques définies sur l'ensemble de valeurs* I*, la taxonomie* $T = (C, \leqslant)$ *et l'ontologie spatio-temporelle* $\mathcal{O}_{\mathrm{st}}$*. La primitive IsInMap* $(m1 : \mathcal{S}, c1 : C)$ *est une primitive telle que* $c1$ *est un concept de la carte* $m1$*. Sa valeur est l'ensemble :* $\{(m1, c1)/\exists n, labelN_{m1}(n) = c1\}$*.*

1. mng_F est une application entre les formules syntaxiques et leur sens.
2. Le produit cartésien indexé par les attributs de A est noté : $\prod_{1 \leqslant i \leqslant n} (Ai : Di)$ ou $\prod(A : D)$

Exemple 32. *Tous les exemples se basent sur l'ensemble de cartes cognitives ontologiques $\mathcal{S}_1$.* `IsInMap(OCM2,?c)` *est une formule primitive. Son sens est l'ensemble des 10 tuples (?c) contenant les concepts de la carte OCM2 :*

$?c$
$PecheCrabe$
$PlaisirPatron$
...

`IsInMap(?m,PecheCrabe)` *est une autre formule primitive. Sa valeur est l'ensemble des 2 tuples (?m) contenant les cartes dans lesquelles le concept PecheCrabe apparaît :*

$?m$
$OCM1$
$OCM2$

Les influences dans une carte cognitive, qui sont les arcs de la carte, ont un effet qui se propage des unes aux autres. Si A influence B et B influence C, on a alors A influence C et cela donne de l'importance aux chemins d'influence du graphe. Ces chemins sont utiles pour déterminer comment un concept est influencé par un autre. Les chemins ont été présentés dans le chapitre 1, il a été choisi de considérer par « chemin », uniquement les chemins minimaux. On appelle le *concept-source* le premier concept de la première influence de la séquence du chemin et *concept-destination* le second concept de la dernière influence de la séquence du chemin.

Path est une primitive qui permet d'accéder aux chemins d'une carte cognitive. Cette primitive relie un chemin à la carte dans laquelle il se trouve ainsi qu'à son concept-source et son concept-destination.

Définition 31 (Primitive Path)**.** *Soit $\mathcal{S}$ un ensemble de cartes cognitives ontologiques définies sur l'ensemble de valeurs I, la taxonomie $T = (C, \leqslant)$ et l'ontologie spatio-temporelle $\mathcal{O}_{\text{st}}$. Soit $Paths_{\mathcal{S}}$ l'ensemble des chemins minimaux de toutes les cartes de $\mathcal{S}$.*
La primitive $Path(m : \mathcal{S}, c1 : C, c2 : C, p : Paths_{\mathcal{S}})$ est une primitive telle que p est un chemin du concept source $c1$ au concept destination $c2$ dans la carte m.
Sa valeur est l'ensemble $\{(m, c1, c2, p) | p \in Paths_m \wedge c1 = source_p \wedge c2 = dest_p\}$.

Exemple 33. `Path(OCM2,DeplacementMer,?c2,?p)` *est une formule primitive. La carte et le concept-source ont été contraints par des constantes, la valeur de la formule primitive est alors l'ensemble des 2 tuples (?c2, ?p) :*

$?c2$	$?p$
$Rentabilite$	$DeplacementMer \rightarrow Rentabilite$
$PlaisirPatron$	$DeplacementMer \rightarrow Rentabilite \rightarrow PlaisirPatron$

`Path(?m,PecheCrabe,Rentabilite,?p)` *est une autre formule primitive. Le concept-source et le concept-destination ont été contraints par des constantes, la valeur de la formule primitive est alors l'ensemble des 2 tuples (?m, ?p) :*

$?m$	$?p$
$OCM2$	$PecheCrabe \rightarrow Rentabilite$
$OCM2$	$PecheCrabe \rightarrow DeplacementMer \rightarrow Rentabilite$

La longueur d'un chemin est le nombre d'influences composant la séquence du chemin. Un chemin A→B→C est composé de deux influences (A,B) et (B,C), sa longueur est alors de 2.

Length est une primitive qui permet d'accéder aux chemins et à leur longueur. Cette primitive relie un chemin à la carte dans laquelle il se trouve et à sa longueur.

Définition 32 (Primitive Length)**.** *Soit $\mathcal{S}$ un ensemble de cartes cognitives ontologiques définies sur l'ensemble de valeurs I, la taxonomie $T = (C, \leqslant)$ et l'ontologie spatio-temporelle $\mathcal{O}_{\text{st}}$. Soit $Paths_{\mathcal{S}}$ l'ensemble des chemins minimaux de toutes les cartes de $\mathcal{S}$.*
La primitive $Length(m : \mathcal{S}, p : Paths_{\mathcal{S}}, l : \mathbb{N})$ est une primitive telle que p est un chemin de longueur l dans la carte m. Sa valeur est l'ensemble : $\{(m, p, l) | p \in Paths_m \wedge length_p = l\}$.

Exemple 34. `Length(?m,?p,1)` *est une formule primitive. La longueur du chemin a été contrainte par une constante, le sens de la formule primitive est alors l'ensemble des 22 tuples (?m, ?p) contenant les chemins de longueur 1 avec les cartes dans lesquelles ils apparaissent :*

$?m$	$?p$
$OCM1$	$PollutionEau \rightarrow ZonePeche$
$OCM1$	$Controle \rightarrow InvestissementNavire$
$OCM2$	$Rentabilite \rightarrow PlaisirPatron$
...	...

Lorsque qu'un concept influence indirectement un second au travers de chemins, il peut être intéressant de savoir quels concepts sont impliqués dans cette influence indirecte. À l'inverse, dans une carte cognitive, il peut être intéressant de savoir dans quels chemins est impliqué un certain concept. Un concept est contenu dans un chemin quand il est présent dans une influence de la séquence du chemin.

Contains est une primitive qui permet d'accéder aux chemins et aux concepts qui sont contenus dans ces chemins. Cette primitive relie un chemin à la carte à laquelle il appartient et aux concepts qu'il contient.

Définition 33 (Primitive Contains)**.** *Soit $\mathcal{S}$ un ensemble de cartes cognitives ontologiques définies sur l'ensemble de valeurs I, la taxonomie $T = (C, \leqslant)$ et l'ontologie spatio-temporelle $\mathcal{O}_{\text{st}}$. Soit $Paths_{\mathcal{S}}$ l'ensemble des chemins minimaux de toutes les cartes de $\mathcal{S}$. $labelN_m$ est l'application d'étiquetage des noeuds de la carte m. La primitive $Contains(m : \mathcal{S}, p : Paths_{\mathcal{S}}, c : C)$*

est une primitive telle que c est un concept faisant partie d'une influence du chemin p dans la carte m. Sa valeur est l'ensemble $\{(m,p,c)|p \in Paths_m \wedge \exists i,(n_i,n_{i+1}) \in p \wedge labelN_m(n_i) = c \vee labelN_m(n_{i+1}) = c\}$

Exemple 35. *`Contains(OCM2,ZonePeche→Rentabilite→PlaisirPatron,?c)` est une formule primitive. La carte et le chemin ont été contraints par des constantes, le sens de la formule primitive est alors l'ensemble des 3 tuples (?c) contenant les concepts du chemin décrit par une constante :*

$?c$
$ZonePeche$
$Rentabilite$
$PlaisirPatron$

`Contains(?m,?p,Rentabilite)` est une autre formule primitive. La concept qui doit appartenir au chemin a été contraint par une constante, le sens de la formule primitive est alors l'ensemble des 15 tuples (?m, ?p) contenant les chemins qui contiennent le concept Rentabilite et les cartes dans lesquelles ils apparaissent :

$?m$	$?c$
$OCM2$	$NavireModerne \rightarrow ZonePeche \rightarrow Rentabilite$
$OCM2$	$ZonePeche \rightarrow Rentabilite \rightarrow PlaisirPatron$
$OCM2$	$SaisonCreuse \rightarrow Rentabilite \rightarrow PlaisirPatron$
...	...

ContainsPath est une primitive similaire qui permet d'accéder à tous les chemins contenus dans un chemin. Un chemin est contenu dans un autre chemin lorsque c'est une séquence qui est une sous-séquence de la séquence du second chemin. Cette primitive relie donc un chemin à la carte à laquelle il appartient et aux chemins qu'il contient.

Définition 34 (Primitive ContainsPath)**.** *Soit $\mathcal{S}$ un ensemble de cartes cognitives ontologiques définies sur l'ensemble de valeurs I, la taxonomie $T = (C, \leqslant)$ et l'ontologie spatio-temporelle $\mathcal{O}_{\text{st}}$. Soit $Paths_{\mathcal{S}}$ l'ensemble des chemins minimaux de toutes les cartes de $\mathcal{S}$. La primitive $ContainsPath(m : \mathcal{S}, p1 : Paths_{\mathcal{S}}, p2 : Paths_{\mathcal{S}})$ est une primitive telle que le chemin $p2$ est une sous-séquence du chemin $p1$ dans la carte m. Sa valeur est l'ensemble :* $\{(m,p1,p2)|p1,p2 \in Paths_m \wedge \exists i < length_{p1}, \forall j <= length_{p2}(u_{i+j},u_{i+j+1}) \in p_1,(v_j,v_{j+1}) \in p_2,(u_{i+j},u_{i+j+1}) = (v_j,v_{j+1})\}$.

Exemple 36. *`ContainsPath(OCM2,ZonePeche→Rentabilite→PlaisirPatron,?p)` est une formule primitive. La carte et le chemin ont été contraints par des constantes, le sens de la formule primitive est alors l'ensemble des 3 tuples (?p) contenant les chemins formant une sous-séquence du chemin décrit en constante :*

$?p$
$ZonePeche \rightarrow Rentabilite$
$Rentabilite \rightarrow PlaisirPatron$
$ZonePeche \rightarrow Rentabilite \rightarrow PlaisirPatron$

`ContainsPath(OCM2,?p,Rentabilite → PlaisirPatron)` est une autre formule primitive. La carte et le chemin ont été contraints par des constantes, le sens de la formule primitive est alors l'ensemble des 8 tuples (?p) contenant les chemins de la carte OCM2 qui contiennent le chemin décrit en constante :

$?p$
$NavireModerne \rightarrow ZonePeche \rightarrow Rentabilite \rightarrow PlaisirPatron$
$ZonePeche \rightarrow Rentabilite \rightarrow PlaisirPatron$
$SaisonCreuse \rightarrow Rentabilite \rightarrow PlaisirPatron$
...

Les influences des cartes cognitives sont étiquetées par une valeur d'influence qui peut renseigner la force de l'influence ou encore sa polarité.

DirectValue est une primitive qui permet d'accéder aux influences et à leur valeur d'influence. Cette primitive relie deux concepts d'un influence à une carte dans laquelle ils apparaissent et à la valeur de l'influence.

Définition 35 (Primitive DirectValue)**.** *Soit $\mathcal{S}$ un ensemble de cartes cognitives ontologiques définies sur l'ensemble de valeurs I, la taxonomie $T = (C, \leqslant)$ et l'ontologie spatio-temporelle $\mathcal{O}_{\text{st}}$. $labelN_m$ et $labelE_m$ sont respectivement les applications d'étiquetage des noeuds et des arcs de la carte m. La primitive $DirectValue(m : \mathcal{S}, c1 : C, c2 : C, i : I)$ est une primitive telle que $c1$ et $c2$ sont deux concepts formant une influence $(c1, c2)$ étiquetée par i dans la carte m. Sa valeur est l'ensemble :*
$\{(m, c1, c2, i) \mid labelE_m((n1, n2)) = i, labelN_m(n1) = c1, labelN_m(n2) = c2\}$

Exemple 37. *`DirectValue(?m,ZonePeche, PlaisirPatron,?i)` est une formule primitive. Les deux concepts de l'influence ont été contraints par des constantes, seulement il n'existe pas d'influence (ZonePeche, PlaisirPatron) ni dans OCM1 ni dans OCM2, le sens de la formule primitive est alors un ensemble vide de tuples (?m, ?i) : $\varnothing$.*

`DirectValue(?m,?c,PlaisirPatron,-1)` est une autre formule primitive. L'influence a été contrainte par sa valeur d'influence et le concept influencé qui sont spécifiés par des constantes, le sens de la formule primitive est alors l'ensemble des 2 tuples (?m, ?c) contenant la carte et les concepts qui ont une influence directe de -1 sur le concept PecheCrabe :

$?m$	$?c$
$OCM1$	$PecheCrabe$
$OCM2$	$PecheCrabe$

L'influence propagée de chemin est une valeur calculée indiquant la valeur d'influence d'un chemin.

PathValue est une primitive qui permet d'accéder aux chemins et à leur valeur. Cette primitive relie un chemin à la carte à laquelle il appartient et à sa valeur d'influence propagée de chemin.

Définition 36 (Primitive PathValue)**.** *Soit $\mathcal{S}$ un ensemble de cartes cognitives ontologiques définies sur l'ensemble de valeurs I, la taxonomie $T = (C, \leqslant)$ et l'ontologie spatio-temporelle $\mathcal{O}_{\text{st}}$. Soit $Paths_{\mathcal{S}}$ l'ensemble des chemins minimaux de toutes les cartes de $\mathcal{S}$. Soit $\mathcal{IP}_{CM}$ l'influence propagée de chemin dans la carte CM. La primitive $PathValue(m : \mathcal{S}, p1 : Paths_{\mathcal{S}}, i : I)$ est une primitive telle que $p1$ est un chemin dans la carte m et son influence propagée de chemin est égale à i. Sa valeur est l'ensemble $\{(m, p1, i) | p1 \in Paths_m \wedge \mathcal{IP}_m(p1) = i\}$.*

Exemple 38. *`PathValue(?m,SaisonCreuse→Rentabilite→PlaisirPatron,?i)` est une formule primitive. La chemin a été contraint par une constante, le sens de la formule primitive est alors l'ensemble de tuples (?m, ?i) contenant la carte dans laquelle se trouve le chemin et la valeur d'influence propagée de ce chemin :*

$?m$	$?i$
$OCM2$	-1

`PathValue(OCM2, ?p, -1)` est une formule primitive. La carte et la valeur d'influence propagée de chemin ont été contraints par des constantes, le sens de la formule primitive est alors l'ensemble de tuples (?p) contenant les chemins ayant une valeur d'influence propagée de chemin égale à -1 dans la carte OCM2 :

$?p$
$PecheCrabe \rightarrow PlaisirPatron$
$SaisonCreuse \rightarrow Rentabilite$
$SaisonCreuse \rightarrow Rentabilite \rightarrow PlaisirPatron$

La primitive *Value* permet d'accéder à l'influence propagée entre deux concepts, cette influence est calculée en agrégeant les valeurs d'influence propagée des chemins entre les deux concepts. La primitive Value met en relation une carte cognitive, deux concepts et la valeur d'influence propagée du premier concept sur le second dans la carte.

Définition 37 (Primitive Value)**.** *Soit $\mathcal{S}$ un ensemble de cartes cognitives ontologiques définies sur l'ensemble de valeurs I, la taxonomie $T = (C, \leqslant)$ et l'ontologie spatio-temporelle $\mathcal{O}_{\text{st}}$. Soit*

$\mathcal{I}_m$ l'influence propagée appliquée à la carte m. La primitive $Value(m : \mathcal{S}, c1 : C, c2 : C, i : I)$ est une primitive telle que l'influence propagée du concept $c1$ sur le concept $c2$ est égale à i dans la carte m. Sa valeur est l'ensemble $\{(m, c1, c2, i) \mid \mathcal{I}_m(c1, c2) = i\}$.

Exemple 39. *`Value(OCM2,PecheCrabe, Rentabilite,?i)` est une formule primitive. La carte et les deux concepts ont été contraints par des constantes, le sens de la formule primitive est alors l'ensemble contenant l'unique tuple (?i) avec l'influence propagée de `PecheCrabe` sur `Rentabilite` dans la carte OCM2 :*

$?i$
0.44

`Value(OCM2, ?c,PlaisirPatron,?i)` est une autre formule primitive. La carte et le second concept ont été contraints par des constantes, le sens de la formule primitive est alors l'ensemble des 2 tuples (?c, ?i) contenant l'influence propagée de tous les concepts sur `PlaisirPatron` dans la carte OCM2 :

$?c$	$?i$
$Rentabilite$	1
$SaisonCreuse$	-1
$NavireModerne$	0.38
$DescriptionMetier$	0
$NavirePolyvalent$	0
...	...

3.2 Primitives d'accès à la composante taxonomique

Cette partie définit les primitives d'accès aux connaissances en lien avec la taxonomie de concepts. La taxonomie organise les concepts en représentant une relation d'ordre partiel de type « est une sorte de » entre les concepts [Cha10 ; CGL09]. Ces concepts peuvent ensuite être utilisés dans les cartes cognitives taxonomiques ou ontologiques. 3 primitives sont définies pour accéder aux connaissances taxonomiques : deux concernant les relations de spécialisation dans la taxonomie et une permettant l'accès à l'influence propagée et à l'influence propagée taxonomique.

KindOf est une primitive qui permet d'accéder aux relations de spécialisation entre les concepts de la taxonomie. Cette primitive relie deux concepts dont le premier est une spécialisation, pas forcément directe, du deuxième concept.

Définition 38 (Primitive KindOf)**.** *Soit $\mathcal{S}$ un ensemble de cartes cognitives ontologiques définies sur l'ensemble de valeurs I, la taxonomie $T = (C, \leqslant)$ et l'ontologie spatio-temporelle $\mathcal{O}_{\text{st}}$. La*

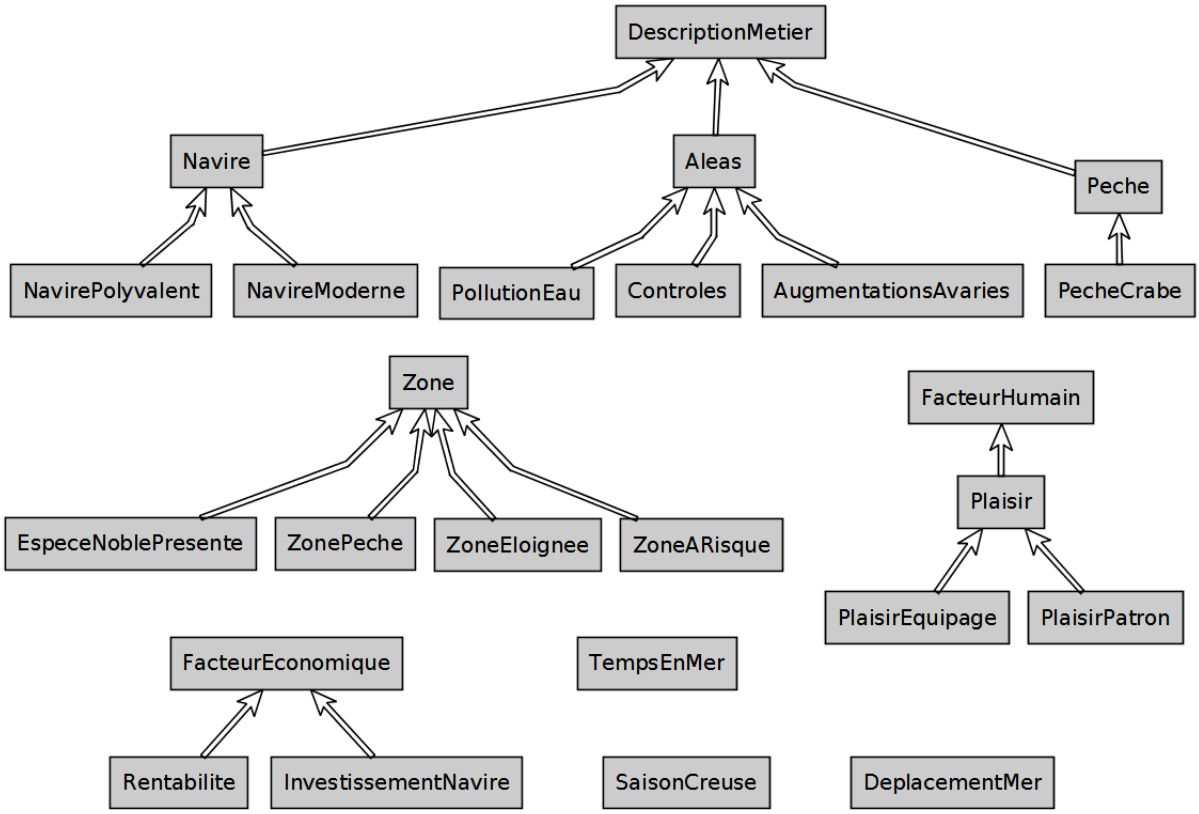


FIGURE 3.2 – Taxonomie T1 de l'ensemble de cartes cognitives ontologiques $\mathcal{S}1$.

primitive $KindOf(c1 : C, c2 : C)$ *est une primitive telle que le concept* $c1$ *est une spécialisation du concept* $c2$ *selon la taxonomie. Sa valeur est l'ensemble* $\{(c1, c2)|c1 \leqslant c2\}$.

Exemple 40. *`KindOf(PlaisirPatron, ?c)` est une formule primitive. Le concept spécialisé a été contraint par la constante `PlaisirPatron`, le sens de la formule primitive est alors l'ensemble des 3 tuples (?c) contenant les concepts généralisant le concept `PlaisirPatron` :*

$?c$
$FacteurHumain$
$Plaisir$
$PlaisirPatron$

`KindOf(?c, FacteurEconomique)` est une autre formule primitive. Le concept général a été contraint par la constante `FacteurEconomique`, le sens de la formule primitive est alors l'ensemble des 3 tuples (?c) contenant les concepts qui sont des spécialisations du concept `FacteurEconomique` :

$?c$
$FacteurEconomique$
$Rentabilite$
$InvestissementNavire$

ElemOf est une primitive qui permet d'accéder aux concepts élémentaires, c'est-à-dire les concepts les plus spécialisés. Elle est particulièrement utile pour accéder à des cartes cognitives contraintes, qui ne contiennent que des concepts élémentaires. Cette primitive relie deux concepts dont le premier est un concept élémentaire et une spécialisation du second concept.

Définition 39 (Primitive ElemOf)**.** *Soit $\mathcal{S}$ un ensemble de cartes cognitives ontologiques définies sur l'ensemble de valeurs I, la taxonomie $T = (C, \leqslant)$ et l'ontologie spatio-temporelle $\mathcal{O}_{\text{st}}$. La primitive $ElemOf(c1 : C, c2 : C)$ est une primitive telle que le concept $c1$ est un concept élémentaire pour le concept $c2$ selon la taxonomie. Sa valeur est l'ensemble :* $\{(c1, c2) \mid c1 \in elemPour_T(c2)\}$.

Exemple 41. *`ElemOf(PlaisirPatron, ?c)` est une formule primitive. Le concept élémentaire a été contraint par la constante `PlaisirPatron`, le sens de la formule primitive est alors l'ensemble des 3 tuples (?c) contenant les concepts dont `PlaisirPatron` est élémentaire :*

$?c$
$FacteurHumain$
$Plaisir$
$PlaisirPatron$

`ElemOf(?c, DescriptionMetier)` est une autre formule primitive. Le concept général a été contraint par la constante `DescriptionMetier`, le sens de la formule primitive est alors l'ensemble des 6 tuples (?c) contenant les concepts élémentaires pour le concept `DescriptionMetier` :

$?c$
$NavirePolyvalent$
$NavireModerne$
$PollutionEau$
$Controle$
$AugmentationAvaries$
$PecheCrabe$

La primitive *TValue* permet d'accéder à l'influence propagée taxonomique [Cha10] entre deux concepts. L'influence propagée taxonomique n'est cependant applicable qu'à des cartes cognitives taxonomiques contraintes, c'est-à-dire quand les cartes ne comportent que des

concepts élémentaires de la taxonomie. La primitive met en relation une carte cognitive, deux concepts et la valeur d'influence propagée taxonomique que le premier concept a sur le second dans la carte.

Définition 40 (Primitive TValue)**.** *Soit $\mathcal{S}$ un ensemble de cartes cognitives ontologiques définies sur l'ensemble de valeurs I, la taxonomie $T = (C, \leqslant)$ et l'ontologie spatio-temporelle $\mathcal{O}_{\text{st}}$. Soit $\mathcal{I}_{T,m}$ l'influence taxonomique appliquée à la carte m. La primitive $TValue(m : \mathcal{S}, c1 : C, c2 : C, i : \mathcal{P}(I))$ est une primitive telle que l'influence propagée taxonomique du concept $c1$ sur le concept $c2$ est égale à i dans la carte m. Sa valeur est l'ensemble $\{(m, c1, c2, i) \| \mathcal{I}_{T,m}(c1, c2) = i\}$.*

Exemple 42. *Les cartes cognitives de l'ensemble $\mathcal{S}_1$ sont contraintes, c'est-à-dire qu'elles ne comportent que des concepts élémentaires de la taxonomie, l'influence propagée taxonomique est alors applicable sur les cartes de $\mathcal{S}_1$.*
`TValue(OCM2,Zone, FacteurEconomique,?i)` *est une formule primitive. La carte et les deux concepts ont été contraints par des constantes, le sens de la formule primitive est alors l'ensemble contenant l'unique tuple (?i) avec l'influence taxonomique de* `Zone` *sur* `FacteurEconomique` *dans la carte OCM2 :*

$?i$
$[0; 0, 5]$

`TValue(?m, PecheCrabe,PlaisirPatron,?i)` *est une autre formule primitive. Les concepts ont été contraints par des constantes, le sens de la formule primitive est alors l'ensemble des 2 tuples (?m, ?i) contenant l'influence taxonomique de* `PecheCrabe` *sur* `PlaisirPatron` *dans les différentes cartes :*

$?m$	$?i$
$OCM1$	-1
$OCM2$	$[-0, 125]$

3.3 Primitives d'accès à la composante temporelle

Le modèle de cartes cognitives ontologiques comporte de nouvelles connaissances par rapport aux modèles de cartes cognitives classiques ou taxonomiques. Ces connaissances se rapportent au temps et à l'espace. Cette partie se focalise sur les connaissances temporelles. Sous forme d'assertions temporelles, ces connaissances se trouvent soit dans l'ontologie temporelle, soit dans les cartes cognitives pour caractériser les noeuds des cartes. Une assertion temporelle est un triplet composé de deux intervalles périodiques, qui sont les entités temporelles utilisées dans cette thèse, et d'un prédicat temporel de comparaison qui permet de mettre

en relation les deux intervalles périodiques. Les noeuds des cartes cognitives ontologiques sont étiquetés par un intervalle périodique qui représente le caractère temporel du noeud, la carte contient un ensemble d'assertions temporelles qui peut alors les mettre en relation avec d'autres intervalles périodiques se trouvant dans l'ontologie. C'est de cette manière que l'on peut caractériser les noeuds des cartes. Cette partie présente deux primitives permettant d'accéder à ces connaissances : la première permet d'accéder aux intervalles périodiques associés aux noeuds des cartes, la seconde permet de comparer les intervalles périodiques en accédant à toutes les assertions temporelles du modèle, implicites comme explicites.

Dans une carte cognitive ontologique, plusieurs noeuds peuvent être étiquetés par le même concept s'ils ont des caractérisations spatio-temporelles différentes. Par exemple la pêche d'un certain type de poisson pourrait influencer la rentabilité à une période de l'année seulement, étant donné que cette espèce de poisson est présente majoritairement à cette période ou qu'il y a une plus grosse demande, et la pêche de ce poisson pourrait influencer le plaisir à un autre moment de l'année car les conditions sont plus agréables. Le concept représentant la pêche de ce poisson pourrait donc étiqueter deux noeuds qui n'influenceraient pas les mêmes concepts. Dans ce cas, la notation de l'intervalle périodique associé à un noeud ne peut se faire seulement par référence à la carte et au concept étiquetant le noeud. La notation utilisée pour un intervalle périodique associé comprend alors :

— un symbole indiquant qu'il s'agit d'un intervalle périodique : « T »

— le nom de la carte cognitive

— le nom du concept étiquetant le noeud en question

— et éventuellement un indice permettant d'identifier le noeud lorsque qu'il partage un concept avec d'autres noeuds.

Notation 2 (Intervalle périodique associé)**.** *Un intervalle périodique associé à un noeud étiqueté par le concept c dans une carte m est noté « T_m_cn » où n est un entier tel qu'il n'existe pas deux noeuds associés au même intervalle périodique. L'identifiant n sera omis lorsque le noeud est le seul à être étiqueté par ce concept dans cette carte.*

Exemple 43. *Supposons que les deux cartes OCM1 (figure 2.11) et OCM2 (figure 3.1) soient une seule et même carte cognitive nommée OCM3. Plusieurs noeuds seraient étiquetés par le même concept, comme ceux étiquetés par le concept* `PecheCrabe`*. Les intervalles périodiques associés à ces noeuds sont notés : « T_OCM3_PecheCrabe1 » et « T_OCM3_PecheCrabe2 ». La plupart des noeuds sont cependant étiquetés par un concept qui n'apparaît qu'un seule fois dans la carte, l'intervalle périodique associé au noeud étiqueté par le concept* `Rentabilite` *est noté « T_OCM3_Rentabilite ».*

TimeInfo est une primitive qui permet d'accéder aux intervalles périodiques associés aux noeuds des cartes. Elle relie un concept à la carte dans laquelle il étiquette un noeud et à l'intervalle périodique qui étiquette le même noeud.

Définition 41 (Primitive TimeInfo)**.** *Soit $\mathcal{S}$ un ensemble de cartes cognitives ontologiques définies sur l'ensemble de valeurs I, la taxonomie $T = (C, \leqslant)$ et l'ontologie spatio-temporelle $\mathcal{O}_{\text{st}}$. Soit $\mathcal{E}_{\text{t}\mathcal{S}}$ l'ensemble des intervalles périodiques associés aux noeuds des cartes de $\mathcal{S}$. La primitive $TimeInfo(m : \mathcal{S}, c1 : C, e : \mathcal{E}_{\text{t}\mathcal{S}})$ est une primitive telle qu'il existe un noeud dans la carte m étiqueté par le concept $c1$ et l'intervalle périodique e. Sa valeur est l'ensemble $\{(m, c1, e) | \exists n, labelN(n) = c1 \wedge labelT(n) = e\}$.*

Exemple 44. *`TimeInfo(OCM2,?c,?t)` est une formule primitive. La carte a été contrainte par une constante, le sens de la formule primitive est alors l'ensemble des 10 tuples (?c, ?t) contenant pour chaque noeud de la carte OCM2 le concept et l'intervalle périodique du noeud :*

$?c$	$?t$
$PecheCrabe$	$T\_OCM2\_PecheCrabe$
$PlaisirPatron$	$T\_OCM2\_PlaisirPatron$
$Rentabilite$	$T\_OCM2\_Rentabilite$
...	...

`TimeInfo(?m, PecheCrabe,?t)` est une autre formule primitive. Le concept qui étiquette le noeud a été contraint par une constante, le sens de la formule primitive est alors l'ensemble des 2 tuples (?m, ?t) contenant les intervalles périodiques associés aux noeuds étiquetés par le concept `PecheCrabe` dans les différentes cartes :

$?m$	$?t$
$OCM1$	$T\_OCM1\_PecheCrabe$
$OCM2$	$T\_OCM2\_PecheCrabe$

Cette primitive a un intérêt limité quand elle est utilisée seule, nous verrons dans le prochain chapitre qu'elle peut être particulièrement utile lorsqu'elle est utilisée conjointement avec la seconde primitive d'accès aux informations temporelles présentée ici : *CompareTime*.

Cette seconde primitive permet l'accès à toutes les assertions temporelles du modèle, qu'elles soient dans une carte ou dans l'ontologie, qu'elles soient explicites ou implicites. Certaines connaissances temporelles sont en effet implicites, par exemple si l'on sait que la saison de la pêche au thon dure un mois, il est implicite qu'elle est plus courte que l'été qui dure trois mois. Pour obtenir ces connaissances implicites, une opération d'inférence est effectuée sur l'ensemble des assertions du modèle. Les connaissances implicites sont inférées à partir d'un ensemble de connaissances explicites, nommé *ensemble de départ*, et d'un ensemble de règles d'inférence. L'application des règles d'inférence à l'ensemble de départ permet de

constituer l'*ensemble stable*, qui contient les connaissances implicites et les connaissances explicites.

De manière générale on définit un *ensemble stable* pour une opération d'inférence : c'est un ensemble dont toutes les connaissances inférables grâce à un ensemble de règles ont été inférées.

Définition 42 (Ensemble stable)**.** *Un* ensemble stable *pour une opération d'inférence est un ensemble pour lequel l'application de l'opération d'inférence ne permet pas de générer de nouveaux éléments.*

Exemple 45. *Prenons un exemple basique pour illustrer la définition d'un ensemble stable. Considérons un ensemble départ E_1 contenant les trois assertions spatiales suivantes :*

- *(Nice, southOf, Lyon)*
- *(Lyon, southOf, Bruxelles)*
- *(Bruxelles, southOf, Amsterdam)*

Considérons également un ensemble de règles R_1 contenant une règle de transitivité et une règle inverse :

- *si (?a, southOf, ?b) et (?b, southOf, ?c) alors (?a, southOf, ?c)*
- *(?a, southOf, ?b) si et seulement si (?b, northOf, ?a)*

L'application des règles de R_1 sur l'ensemble de départ E_1 permet d'obtenir 9 nouvelles assertions :

- *(Nice, southOf, Bruxelles)*
- *(Lyon, southOf, Amsterdam)*
- *(Nice, southOf, Amsterdam)*
- *(Amsterdam, northOf, Bruxelles)*
- *(Amsterdam, northOf, Lyon)*
- *(Amsterdam, northOf, Nice)*
- *(Bruxelles, northOf, Lyon)*
- *(Bruxelles, northOf, Nice)*
- *(Lyon, northOf, Nice)*

L'ensemble constitué des trois assertions de E_1 et des neuf assertions inférées est un ensemble stable pour l'opération d'inférence appliquant les règles de R_1. En effet, les deux règles de R_1 ne permettent pas d'inférer de nouvelles assertions.

Les règles d'inférence utilisées dans cette thèse sont des règles SWRL [Hor+04]. SWRL est un langage de règles pour le web sémantique qui est basé sur OWL et RuleML. Une règle SWRL est un axiome OWL qui a la forme générale : « Antécédent $\Longrightarrow$ Conséquent » où l'antécédent et le conséquent sont des conjonctions d'atomes. Les atomes peuvent être par exemple des assertions de classes, des assertions de propriétés ou des expressions numériques.

Les règles présentées dans ce chapitre utilisent exclusivement les atomes de type assertion de propriétés. Les antécédents comportent un ou deux atomes et les conséquents ne comportent qu'un seul atome. Les règles utilisées peuvent prendre trois formes présentées ci-dessous selon la syntaxe formelle EBNF [Hor+04 ; Pat13] :

— Implies(Antecedent(p1(I-variable(x1) I-variable(x2))
p2(I-variable(x2) I-variable(x3)))
Consequent(p3(I-variable(x1) I-variable(x3))))

— Implies(Antecedent(p1(I-variable(x1) I-variable(x2))
Consequent(p2(I-variable(x1) I-variable(x2))))

— Implies(Antecedent(p1(I-variable(x1) I-variable(x2))
Consequent(p2(I-variable(x2) I-variable(x1))))

De manière moins formelle mais plus lisible, ces trois formes de règles sont présentées en syntaxe HRS [Hor+04] :

— $p1(?x1, ?x2) \wedge p2(?x2, ?x3) \Longrightarrow p3(?x1, ?x3)$

— $p1(?x1, ?x2) \Longrightarrow p2(?x1, ?x2)$

— $p1(?x1, ?x2) \Longrightarrow p2(?x2, ?x1)$

p1,p2 et p3 sont des prédicats alors que x1,x2 et x3 sont des variables. `p1(?x1,?x2)`, qui désigne une assertion de propriété, est une notation équivalente à `(?x1,p1,?x2)`, qui désigne une assertion spatio-temporelle. La première forme de règle est connue sous le nom de « règle de l'oncle » [Krö+11] car elle permet de définir une relation d'oncle en remplaçant p1,p2 et p3 respectivement par hasParent, hasBrother et hasUncle, cette forme généralise également les règles de transitivité de propriété. La deuxième forme de règle décrit des règles de sous-propriété, pour exprimer par exemple que la propriété hasBrother est une sous-propriété de hasSibling. La troisième forme de règle décrit des règles de propriété inverse, pour exprimer par exemple que hasChild est une relation inverse de hasParent. Cette troisième forme généralise les règles de propriété symétrique, qui permettent d'exprimer par exemple que la propriété

hasSibling est symétrique.

179 règles d'inférences sont utilisées pour les assertions temporelles. Les règles sont décrites brièvement ici mais données exhaustivement en annexe. La plupart de ces règles, 148, concernent les relations topologiques de Balbiani et Osmani [Osm99], 16 de ces 148 règles sont des règles inverse la forme p1(?x1, ?x2) $\implies$ p2(?x2, ?x1), toutes les autres (132) sont de la forme p1(?x1, ?x2) $\wedge$ p2(?x2, ?x3) $\implies$ p3(?x1, ?x3) et proviennent de la table de composition proposée par les mêmes auteurs [BO00].

Exemple 46. *Les exemples donnés dans les prochains exemples se basent sur l'ontologie temporelle $\mathcal{O}_{\mathrm{t1}}$ (figure 2.5).*

— *La règle $mmi(?x1, ?x2) \implies mmi(?x2, ?x1)$ est utilisée pour inférer l'assertion temporelle (Ete, mmi, SaisonCrabe) à partir de l'assertion (SaisonCrabe, mmi, Ete). Elle signifie que l'été finit au début de la saison du crabe et commence quand la saison du crabe se termine.*

— *La règle $ppi(?x1, ?x2) \wedge mmi(?x2, ?x3) \implies d(?x1, ?x3)$ est utilisée pour inférer l'assertion temporelle (Hiver, d,SaisonCrabe) à partir des assertions (Hiver, ppi, Ete) et (Ete, mmi, SaisonCrabe). Elle signifie que l'hiver est pendant la saison du crabe.*

Les prédicats temporels In (Inside) et Dis (Disjoint) ont été définis dans le chapitre précédent comme la disjonction de quatre prédicats temporels, respectivement de eq(equals), s(starts), d(during), f(finishes) et de m(meets), mi(is met), mmi (meet and is met), ppi (precedes and is preceded). 8 règles de la forme p1(?x1, ?x2) $\implies$ p2(?x1, ?x2) concernent ces deux prédicats, par exemple le fait qu'un intervalle périodique en commence un autre (prédicat starts) implique qu'il soit contenu dans cet intervalle (Inside).

Exemple 47. — *La règle $s(?x1, ?x2) \implies in(?x1, ?x2)$ est utilisée pour inférer l'assertion temporelle (Printemps, in,SaisonThon) à partir de l'assertion (Printemps, s, SaisonThon). Elle signifie que le printemps est compris dans la saison du thon.*

13 règles concernent les prédicats temporels de comparaison de durées dont 3 règles inverse, 3 règles de la forme p1(?x1, ?x2) $\implies$ p2(?x1, ?x2) qui permettent d'inférer par exemple qu'un intervalle périodique est de plus courte durée qu'un second s'il le termine, et 7 règles provenant de la table de composition suivante :

	$=_t$	$<_t$	$>_t$
$=_t$	$=_t$	$<_t$	$>_t$
$<_t$	$<_t$	$<_t$	
$>_t$	$>_t$		$>_t$

La table de composition est une manière simple de représenter les règles du type :
p1(?x1, ?x2) $\wedge$ p2(?x2, ?x3) $\implies$ p3(?x1, ?x3).

Par exemple la case à la première ligne et la deuxième colonne représente la règle :
$=_t$(e1, e2) $\wedge$ $<_t$(e2, e3) $\implies$ $<_t$(e1, e3).

Exemple 48. — *La règle* $<_t (?x1, ?x2) \implies >_t (?x2, ?x1)$ *est utilisée pour inférer l'assertion temporelle (Mois,* $>_t$*,Semaine) à partir de l'assertion (Semaine,* $<_t$*, Mois).*

Les règles sont présentées en SWRL, elles auraient pu être présentées en tant qu'expressions de propriété OWL 2. En effet, chacune des règles présentées peut être indépendamment exprimée en OWL 2 au travers des expressions de propriétés [Hit+09] données ici au format *Functional-Style Syntax*[3] :

— La règle : p1(?x1, ?x2) $\wedge$ p2(?x2, ?x3) $\implies$ p3(?x1, ?x3) est exprimable par des propriétés transitives si p1=p2=p3 :
`TransitiveObjectProperty( :p1 )`,
sinon par des inclusions complexes de rôles :
`SubObjectPropertyOf( ObjectPropertyChain( :p1 :p2 ) :p3 )`

— La règle p1(?x1, ?x2) $\implies$ p2(?x1, ?x2) est exprimable par des sous-propriétés :
`SubObjectPropertyOf( :p1 :p2 )`

— La règle p1(?x1, ?x2) $\implies$ p2(?x2, ?x1) est exprimable par des propriétés symétriques si p1=p2 :
`SymmetricObjectProperty( :p1 )`,
sinon par des propriétés inverses :
`InverseObjectProperties( :p1 :p2 )`

Cependant, l'utilisation de ces expressions de propriétés requiert certaines contraintes, notamment celle imposant un ordre partiel strict sur les propriétés [HKS06]. Cet ordre strict n'est pas respecté par les règles présentées, par exemple les deux règles suivantes rendent tout ordre strict impossible :

— (1) mmi(?x1, ?x2) $\wedge$ mmi(?x2, ?x3) $\implies$ eq(?x1, ?x3)

— (2) mmi(?x1, ?x2) $\wedge$ eq(?x2, ?x3) $\implies$ mmi(?x1, ?x3)

En effet, la règle (1) implique l'ordre mmi < eq et la règle (2) implique l'ordre eq < mmi, le problème d'inférence appliquant ces règles est alors indécidable et n'est pas pris en compte par OWL. C'est pour cela que les règles présentées sont exprimées en SWRL.

L'*ensemble stable des assertions temporelles* est l'ensemble de toutes les assertions temporelles du modèle, celles explicites comme celles implicites. C'est l'ensemble stable des assertions temporelles pour l'opération d'inférence appliquant les 179 règles présentées précédemment à partir de l'ensemble de départ étant l'union des différents ensembles d'assertions temporelles des différentes cartes et de l'ontologie.

3. `https://www.w3.org/TR/owl2-syntax/#Functional-Style_Syntax`

Définition 43 (Ensemble stable des assertions temporelles)**.** *Soit $\mathcal{S}$ un ensemble de n cartes cognitives ontologiques définies sur l'ensemble de valeurs I, la taxonomie $T = (C, \leqslant)$ et l'ontologie spatio-temporelle $\mathcal{O}_{\mathrm{st}}$. Soit $\mathcal{R}_t$ l'ensemble des 179 règles temporelles d'inférence. Soit $\mathcal{A}_t$ l'ensemble des assertions temporelles de l'ontologie spatio-temporelle et $\mathcal{A}_{ti}$ les assertions temporelles de la carte i.*

L'ensemble stable des assertions temporelles $\mathcal{AS}_{\mathrm{t}}$ est l'ensemble stable pour l'application des règles de $\mathcal{R}_t$ à partir de l'ensemble de départ : $\mathcal{A}_t \cup \bigcup_{i=1}^{n} \mathcal{A}_{ti}$

CompareTime est une primitive qui permet d'accéder à l'ensemble stable des assertions temporelles. Cette primitive relie deux intervalles périodiques et prédicat temporel de comparaison tels qu'ils forment une assertion temporelle valide faisant partie de l'ensemble stable des assertions temporelles.

Définition 44 (CompareTime)**.** *Soit $\mathcal{S}$ un ensemble de cartes cognitives ontologiques définies sur l'ensemble de valeurs I, la taxonomie $T = (C, \leqslant)$ et l'ontologie spatio-temporelle $\mathcal{O}_{\mathrm{st}}$. Soit $\mathcal{E}_{\mathrm{t}}$ l'union de l'ensemble des intervalles périodiques associés aux noeuds des cartes de $\mathcal{S}$ et de celui des intervalles périodiques de l'ontologie $\mathcal{O}_{\mathrm{st}}$. Soit $\mathcal{P}_{\mathrm{t}}$ l'ensemble des prédicats temporels de comparaison. Soit $\mathcal{AS}_{\mathrm{t}}$ l'ensemble stable des assertions temporelles de $\mathcal{S}$. La primitive $CompareTime(e1 : \mathcal{E}_{\mathrm{t}}, p : \mathcal{P}_{\mathrm{t}}, e2 : \mathcal{E}_{\mathrm{t}})$ est une primitive telle que les deux intervalles périodiques et le prédicat forment une assertion temporelle faisant partie de l'ensemble stable des assertions temporelles. Sa valeur est l'ensemble $\mathcal{AS}_{\mathrm{t}}$.*

Exemple 49. *Cet exemple se base sur l'ensemble de cartes cognitives ontologiques $\mathcal{S}$ et utilise l'ontologie temporelle de la figure 2.5.*

`CompareTime(T_OCM1_PecheCrabe,?p,T_OCM2_PecheCrabe)` *est une formule primitive. Les intervalles périodiques ont été contraints par des constantes : le premier est celui associé au noeud qui est étiqueté par le concept PecheCrabe dans la carte OCM1, le second est celui associé au noeud qui est étiqueté par le concept PecheCrabe dans la carte OCM2. Le sens de la formule primitive est alors l'ensemble des 3 tuples (?p) contenant les prédicats permettant de comparer les intervalles périodiques spécifiés :*

$?p$
$finishes$
$inside$
$<_t$

Ces trois tuples nous indiquent que le pêcheur de la carte 1 pêche le crabe à la fin de la période à laquelle le second pêcheur pèche le crabe, et par conséquent pendant cette période et pour une période plus courte. Ces trois assertions temporelles ont été obtenues grâce au processus d'inférence. En effet les deux assertions (T_OCM1_PecheCrabe, finishes, SaisonCrabe) et (T_OCM2_PecheCrabe, equals, SaisonCrabe) nous ont permis d'obtenir ces nouvelles connaissances en utilisant les règles :

— *equals(?e1, ?e2)* $\Longrightarrow$ *equals(?e2, ?e1)*

— *finishes(?e1, ?e2)* $\wedge$ *equals(?e2, ?e3)* $\Longrightarrow$ *finishes(?e1, ?e3)*

— *finishes(?e1, ?e2)* $\Longrightarrow$ *inside(?e1, ?e2)*

— *finishes(?e1, ?e2)* $\Longrightarrow$ $<_t$*(?e1, ?e2)*

`CompareTime(T_OCM2_SaisonCreuse, =t ,?e2)` *est une autre formule primitive. Le premier intervalle périodique et le prédicat temporel ont été contraints par des constantes, le sens de la formule primitive est alors l'ensemble des 6 tuples (?e2) contenant les intervalles périodiques ayant la même durée que celui spécifié :*

$?e2$
Ete
$Saison$
$Hiver$
...

Ces tuples nous indiquent la durée de la saison creuse pour le second pêcheur (carte OCM2). Mis à part le premier tuple (Ete) qui provient d'une assertion temporelle de la carte OCM2, les 5 autres tuples ont été inférés grâce aux règles suivantes :

— *equals(?e1, ?e2)* $\Longrightarrow$ $=_t$*(?e1, ?e2)*

— $=_t$*(?e1, ?e2)* $\wedge$ $=_t$*(?e2, ?e3)* $\Longrightarrow$ $=_t$*(?e1, ?e3)*

3.4 Primitives d'accès à la composante spatiale

De manière similaire à la partie précédente, cette partie se focalise sur les connaissances spatiales. Sous forme d'assertions spatiales, ces connaissances se trouvent soit dans l'ontologie spatiale, soit dans les cartes cognitives pour caractériser les noeuds des cartes. Une assertion spatiale est un triplet composé de deux entités spatiales et d'un prédicat spatial de comparaison qui permet de mettre en relation les deux entités spatiales. Les noeuds des cartes cognitives ontologiques sont étiquetés par une entité spatiale qui représente le caractère spatial du noeud, la carte contient un ensemble d'assertions spatiales qui peut alors les mettre en relation avec d'autres entités se trouvant dans l'ontologie. C'est de cette manière que l'on peut caractériser les noeuds des cartes. Cette partie présente deux primitives permettant d'accéder à ces connaissances : la première permet d'accéder aux entités spatiales associées aux noeuds des cartes, la seconde permet de comparer les entités spatiales en accédant à toutes les assertions spatiales du modèle, implicites comme explicites.

Nous avons vu que dans une carte cognitive ontologique, plusieurs noeuds peuvent être étiquetés par le même concept s'ils ont des caractérisations spatio-temporelles différentes.

Pour cela, la notation de l'intervalle périodique associé à un noeud avait du être adaptée, c'est également le cas pour l' entité spatiale associée à un noeud. Dans ce cas le symbole préfixant le nom de l'entité spatiale est « S » à la place de « T » pour les intervalles périodiques.

Notation 3 (Entité spatiale associée)**.** *Une entité spatiale associée à un noeud étiqueté par le concept c dans une carte m est notée « S_m_cn » où n est un entier tel qu'il n'existe pas deux noeuds associés à la même entité spatiale. L'identifiant n sera omis lorsque le noeud est le seul à être étiqueté par ce concept dans cette carte.*

Exemple 50. `S_OCM2_Rentabilite` *et* `S_OCM2_ZonePeche` *sont des entités spatiales associées à des concepts de la carte OCM2, respectivement aux concepts Rentabilite et ZonePeche.*

SpaceInfo est une primitive qui permet d'accéder aux entités spatiales associées aux noeuds des cartes. Elle relie un concept à la carte dans laquelle il étiquette un noeud et à l'entité spatiale qui étiquette le même noeud.

Définition 45 (Primitive SpaceInfo)**.** *Soit $\mathcal{S}$ un ensemble de cartes cognitives ontologiques définies sur l'ensemble de valeurs I, la taxonomie $T = (C, \leqslant)$ et l'ontologie spatio-temporelle $\mathcal{O}_{\text{st}}$. Soit $\mathcal{E}_{\text{s}\mathcal{S}}$ l'ensemble des entités spatiales associés aux noeuds des cartes de $\mathcal{S}$. La primitive $SpaceInfo(m : \mathcal{S}, c1 : C, e : \mathcal{E}_{\text{s}\mathcal{S}})$ est une primitive telle qu'il existe un noeud dans la carte m étiqueté par le concept $c1$ et l'entité spatiale e. Sa valeur est l'ensemble : $\{(m, c1, e) | \exists n, labelN(n) = c1 \wedge labelS(n) = e\}$.*

Exemple 51. `SpaceInfo(OCM2,?c,?s)` *est une formule primitive. La carte a été contrainte par une constante, le sens de la formule primitive est alors l'ensemble des 10 tuples (?c, ?s) contenant pour chaque noeud de la carte OCM2, le concept et l'entité spatiale du noeud :*

$?c$	$?s$
$PecheCrabe$	$S\_OCM2\_PecheCrabe$
$PlaisirPatron$	$S\_OCM2\_PlaisirPatron$
$Rentabilite$	$S\_OCM2\_Rentabilite$
...	...

`SpaceInfo(?m, PecheCrabe,?s)` *est une autre formule primitive. Le concept qui étiquette le noeud a été contraint par une constante, le sens de la formule primitive est alors l'ensemble des 2 tuples (?m, ?s) contenant les entités spatiales associées aux noeuds étiquetés par* `PecheCrabe` *dans les différentes cartes :*

$?m$	$?s$
$OCM1$	$S\_OCM1\_PecheCrabe$
$OCM2$	$S\_OCM2\_PecheCrabe$

On remarque que les noeuds étiquetés par le concept PecheCrabe dans les deux cartes ne sont pas caractérisés spatialement, ils sont tout de même étiquetés par une entité spatiale comme chaque noeud d'une carte.

Tout comme la primitive *TimeInfo*, la primitive *SpaceInfo* a un intérêt limité quand elle est utilisée seule et non avec la seconde primitive d'accès aux informations spatiales : *CompareSpace*. Cette seconde primitive permet l'accès à toutes les assertions spatiales du modèle. De manière analogue à la partie temporelle, nous présentons ici les règles d'inférence utilisées sur les connaissances spatiales.

80 règles d'inférences sont utilisées pour les assertions spatiales, elles sont de la même forme celles utilisées pour les assertions temporelles. Une bonne partie de ces règles, 35, concernent les relations topologiques RCC [Coh+97 ; RC89 ; RCC92], 8 de ces 35 règles sont de la forme p1(?x1, ?x2) $\implies$ p2(?x2, ?x1), les 27 autres sont de la forme p1(?x1, ?x2)$\wedge$ p2(?x2, ?x3) $\implies$ p3(?x1, ?x3) et proviennent de la table de composition proposée par différents auteurs [LY03].

Exemple 52. *Les exemples donnés dans les prochains exemples se basent sur l'ontologie spatiale $\mathcal{O}_{\mathrm{s1}}$ (figure 2.9).*

— *La règle $EC(?x1, ?x2) \implies EC(?x2, ?x1)$ est utilisée pour inférer l'assertion spatiale (Port, EC, Mer) à partir de l'assertion (Mer, EC, Port). Elle signifie que le port est connecté à la mer.*

Concernant les prédicats spatiaux de comparaison de distances : {very_close, close, far, very_far}, seulement 4 règles de la forme p1(?x1, ?x2) $\implies$ p2(?x2, ?x1) ont été ajoutées permettant d'inférer des assertions comme (B,far,A) si (A,far,B). Concernant les prédicats spatiaux de comparaison d'orientation, représentant les 8 points cardinaux, 8 règles de la forme p1(?x1, ?x2) $\implies$ p2(?x2, ?x1) et 8 règles de la forme p1(?x1, ?x2)$\wedge$ p2(?x2, ?x3) $\implies$ p3(?x1, ?x3) ont été ajoutées. Elles correspondent respectivement aux propriétés inverses et à la transitivité de chaque prédicat. Pour les prédicats spatiaux de comparaison d'orientation, comme ceux de distances, on peut être tenté d'ajouter des règles pour inférer par exemple que A est très proche de C quand A est très proche de B et B de C. Ces règles n'ont pas été ajoutées car elles pourraient conduire à des connaissances fausses, en effet ces relations qualitatives sont flexibles dans leur sens et requièrent donc de ne pas surestimer leur exactitude et précision lors du processus d'inférence.

Exemple 53.
— *La règle $southOf(?x1, ?x2) \implies northOf(?x2, ?x1)$ est utilisée pour inférer l'assertion spatiale (StNazaire, northOf,Pornic) à partir de l'assertion (Pornic, southOf, StNazaire).*

— *La règle $southOf(?x1, ?x2) \wedge southOf(?x2, ?x3) \implies southOf(?x1, ?x3)$ est utilisée pour savoir que l'île d'Yeu se situe au sud de Pornic en inférant l'assertion spatiale (Yeu, sou-*

thOf,Pornic) à partir des assertions (Yeu, southOf, Herbodiere) et (Herbodiere, southOf, Pornic).

Les prédicats spatiaux de comparaison de surfaces ($<_s, =_s, >_s$) et de profondeur ou d'altitude (is_lower, has_height,is_higher) sont très similaires aux prédicats temporels de comparaison de durées. Ils ont 10 règles similaires les concernant, dont 3 de la forme p1(?x1, ?x2) $\Longrightarrow$ p2(?x2, ?x1), et 7 de la forme p1(?x1, ?x2) $\wedge$ p2(?x2, ?x3) $\Longrightarrow$ p3(?x1, ?x3) qui proviennent des tables de composition suivantes :

	$=_s$	$<_s$	$>_s$
$=_s$	$=_s$	$<_s$	$>_s$
$<_s$	$<_s$	$<_s$	
$>_s$	$>_s$		$>_s$

	hh	il	ih
hh	hh	il	ih
il	il	il	
ih	ih		ih

où hh,il et ih sont respectivement les prédicats has_height, is_lower et is_higher. Les prédicats de comparaison de surfaces, comme les prédicats de comparaison de durées ont des règles de la forme p1(?x1, ?x2) $\Longrightarrow$ p2(?x1, ?x2) supplémentaires permettant d'inférer par exemple qu'une entité est plus petite qu'une seconde qui la contient, il y en a 5 pour les prédicats concernant les surfaces.

Exemple 54. — *La règle $TPP(?x1, ?x2) \Longrightarrow <_s (?x1, ?x2)$ est utilisée pour inférer l'assertion spatiale (ZoneCotiere, $<_s$,Mer) à partir de l'assertion (ZoneCotiere, TPP, Mer).*

A l'inverse des règles portant sur le temps, les 80 règles portant sur l'espace sont exprimables en OWL 2 tout en préservant un ordre partiel strict sur les propriétés. Elles sont ici exprimées en SWRL par souci d'homogénéisation.

L'*ensemble stable des assertions spatiales*, semblable à l'ensemble stable des assertions temporelles, est l'ensemble de toutes les assertions spatiales du modèle, celles explicites comme celles implicites. C'est l'ensemble stable des assertions spatiales pour l'opération d'inférence appliquant les 80 règles présentées précédemment à partir de l'ensemble de départ qui est l'union des différents ensembles d'assertions spatiales des différentes cartes et de l'ontologie.

Définition 46 (Ensemble stable des assertions spatiales)**.** *Soit $\mathcal{S}$ un ensemble de n cartes cognitives ontologiques définies sur l'ensemble de valeurs I, la taxonomie $T = (C, \leqslant)$ et l'ontologie spatio-temporelle $\mathcal{O}_{\mathrm{st}}$. Soit $\mathcal{R}_s$ l'ensemble des 80 règles spatiales d'inférence. Soit $\mathcal{A}_s$ l'ensemble des assertions spatiales de l'ontologie spatio-temporelle et $\mathcal{A}_{si}$ les assertions spatiales de la carte i.*

L'ensemble stable des assertions spatiales $\mathcal{AS}_{\mathrm{s}}$ est l'ensemble stable pour l'application des règles de $\mathcal{R}_s$ à partir de l'ensemble de départ : $\mathcal{A}_s \cup \bigcup_{i=1}^{n} \mathcal{A}_{si}$

CompareSpace est une primitive qui permet d'accéder à l'ensemble stable des assertions spatiales. Cette primitive relie deux entités spatiales et un prédicat spatial de comparaison tels qu'ils forment une assertion spatiale valide faisant partie de l'ensemble stable des assertions spatiales.

Définition 47 (CompareSpace)**.** *Soit $\mathcal{S}$ un ensemble de cartes cognitives ontologiques définies sur l'ensemble de valeurs I, la taxonomie $T = (C, \leqslant)$ et l'ontologie spatio-temporelle $\mathcal{O}_{\mathrm{st}}$. Soit $\mathcal{E}_{\mathrm{s}}$ l'union de l'ensemble des entités spatiales associés aux noeuds des cartes de $\mathcal{S}$ et de celui des entités spatiales de l'ontologie $\mathcal{O}_{\mathrm{st}}$. Soit $\mathcal{P}_{\mathrm{s}}$ l'ensemble des prédicats spatiaux de comparaison. Soit $\mathcal{AS}_{\mathrm{s}}$ l'ensemble stable des assertions spatiales de $\mathcal{S}$. La primitive $CompareSpace(e1 : \mathcal{E}_{\mathrm{s}}, p : \mathcal{P}_{\mathrm{s}}, e2 : \mathcal{E}_{\mathrm{s}})$ est une primitive telle que les deux entités spatiales et le prédicat forment une assertion spatiale faisant partie de l'ensemble stable des assertions spatiales. Sa valeur est l'ensemble $\mathcal{AS}_{\mathrm{s}}$.*

Exemple 55. *Cet exemple se base sur l'ensemble de cartes cognitives ontologiques $\mathcal{S}$ et utilise l'ontologie spatiale de la figure 2.9.*

`CompareSpace(S_OCM1_ZonePeche,?p,S_OCM2_ZonePeche)` *est une formule primitive. Les entités spatiales ont été contraintes par des constantes : la première est celle associée au noeud qui est étiqueté par le concept ZonePeche dans la carte OCM1, la seconde est celle associée au noeud qui est étiqueté par le concept ZonePeche dans la carte OCM2. Le sens de la formule primitive est alors l'ensemble contenant l'unique tuple (?p) contenant l'unique présicat pouvant comparer les deux entités spatiales spécifiées :*

$?p$
$<_s$

Ce tuple nous indique que la zone de pêche du pêcheur de la carte OCM1 est plus petite que celle du second pêcheur de la carte OCM2. Cette assertion spatiale a été obtenue grâce au processus d'inférence. En effet les assertions (S_OCM1_ZonePeche, $<_s$, petiteZonePeche), (petiteZonePeche, $<_s$ GrandeZonePeche) et (S_OCM2_PecheCrabe, $=_s$, GrandeZonePeche) nous ont permis d'obtenir cette nouvelle assertion spatiale en utilisant les règles :

- *$=_s$(?e1, ?e2) $\Longrightarrow$ $=_s$(?e2, ?e1)*
- *$<_s$(?e1, ?e2) $\wedge$ $<_s$(?e2, ?e3) $\Longrightarrow$ $<_s$(?e1, ?e3)*
- *$<_s$(?e1, ?e2) $\wedge$ $=_s$(?e2, ?e3) $\Longrightarrow$ $<_s$(?e1, ?e3)*

`CompareSpace(S_OCM2_ZoneARisque, ?p ,?e2)` *est une autre formule primitive. Une entité spatiale a été contrainte par une constante, le sens de la formule primitive est alors l'ensemble des 2 tuples (?p, ?e2) contenant les prédicats et entités spatiales comparables à l'entité spécifiée :*

$?p$	$?e2$
$southOf$	$Pornic$
$southOf$	$StNazaire$

Ces deux tuples nous indiquent la direction de la zone à risque pour le second pêcheur (carte OCM2). Le premier tuple (southOf,Pornic) provient d'une assertion spatiale de la carte OCM2, le second tuple a été inféré grâce à la règle suivante :

— *southOf(?e1, ?e2)* $\wedge$ *southOf(?e2, ?e3)* $\Longrightarrow$ *southOf(?e1, ?e3)*

Les primitives d'accès aux connaissances temporelles et aux connaissances spatiales sont très similaires, cela n'est pas étonnant étant donné que la structure de ces connaissances est la même sous forme d'assertions. La principale différence vient du sens des entités mais aussi du sens des prédicats et donc des règles d'inférences. On peut se poser diverses questions sur ces ensembles de règles ainsi que sur l'opération d'inférence, par exemple concernant la complétude des ensembles de règles, la complexité de l'inférence ou sa tractabilité.
Concernant la complétude l'ensemble de règles, elle n'est pas garantie : en effet certaines règles ont été omises comme celles inférant des disjonctions d'assertions qui ne sont pas prises en compte dans le modèle proposé. Concernant la complexité, les problèmes d'inférence classiques sont indécidables en SWRL [Hor+05 ; Hor05b]. Une solution pour obtenir un ensemble de règles permettant une inférence décidable serait de supprimer quelques règles, pour obtenir un ensemble utilisable en OWL 2 qui est décidable mais tout de même coûteux (au moins NExpTime). Bien que la complexité théorique des ensembles de règles soit trop élevée, ces règles ont été testées sur un ensemble de cartes cognitives provenant du cas réel qu'est le projet Kifanlo. Grâce aux primitives d'accès, il a été possible d'accéder à l'ensemble stables des assertions spatiales et temporelles en un temps raisonables, de l'ordre de quelques secondes pour obtenir plus de 2000 assertions inférées.

Conclusion

Ce chapitre propose l'utilisation de primitives afin d'accéder aux diverses connaissances du modèle des cartes cognitives ontologiques : ces connaissances qu'elles concernent les cartes, leurs influences, la taxonomie et l'ontologie spatio-temporelle. Les primitives sont des relations entre les objets du modèle et permettent de représenter les connaissances de manière structurée. Une formule primitive est une expression syntaxique de primitive, elle permet de contraindre la valeur d'une formule primitive par le biais de constantes et de variables, et ce faisant d'accéder aux connaissances désirées. Un des avantages des primitives est qu'elles peuvent faire face aux évolutions du modèle, en effet l'ajout de primitives est toujours possible pour accéder aux différentes extensions du modèle. Par exemple dans nos travaux, les primitives ont été proposées initialement sur le modèle des cartes cognitives taxonomiques

et d'autres primitives ont été proposées lorsqu'on s'est intéressé au modèle des cartes cognitives ontologiques. Certaines primitives peuvent être définies par une opération d'inférence afin d'obtenir leur valeur, pour cela des règles d'inférences sont proposées, notamment pour les primitives CompareTime et CompareSpace. Des travaux supplémentaires pourraient être effectuées concernant la complexité de l'inférence, ou l'amélioration des problèmes d'optimisation et de complétude. Ces primitives ont néanmoins été testées dans le logiciel VSPCC sur un ensemble de cartes cognitives ontologiques provenant d'un cas concret d'utilisation [4], l'accès à l'ensemble des connaissances spatio-temporelles n'a pas posé de problème particulier. Les primitives posent un fondement pour accéder aux connaissances du modèle. Le chapitre suivant présente le langage de requête CMQL qui s'appuie sur les primitives présentées dans ce chapitre.

4. Ces cartes sont les 53 du corpus du projet Kifanlo présenté dans les chapitres 1 et 5.

CHAPITRE 4

CMQL : UN LANGAGE D'INTERROGATION DE CARTES COGNITIVES ONTOLOGIQUES

Introduction

Les primitives d'accès permettent de structurer et d'accéder directement aux différents objets du modèle des cartes cognitives ontologiques. L'accès aux connaissances contenues dans un modèle de représentation de connaissances se doit d'être précis et exhaustif. En effet, représenter des connaissances sert principalement à les structurer pour raisonner dessus et les restituer.

Une manière simple d'interroger des connaissances est d'exprimer ce que l'on veut obtenir au travers d'un langage de requête déclaratif. De nombreux langages de requête ont été conçus pour accéder aux connaissances de différents modèles de connaissances. La conception d'un langage dépend évidemment du modèle qu'il permet d'interroger ; étant donnée la grande variété des modèles, ces langages sont différents mais peuvent présenter des similitudes. SQL nous intéresse particulièrement car il très connu en dehors de la sphère des informaticiens et notamment par le géographe utilisateur à l'origine du projet Kifanlo. Le langage SQL, qui sert à interroger des connaissances structurées selon le modèle relationnel, est un des langages de requête les plus utilisés, il a une grosse influence sur la plupart des langages de requêtes conçus ultérieurement. Deux propositions de langage de requêtes sur des modèles graphiques sont importantes à citer. Le langage SPARQL [PS06], qui interroge les modèles *OWL/RDF* [AV04 ; MMM+04 ; MV+04], s'inspire de SQL et permet l'interrogation de connaissances RDF par la description de patrons de graphes. Le langage Cypher [Ang+17 ; LMD14], qui interroge le modèle des *graphes de propriétés* [Ang18], est un des langages les plus utilisés pour ces graphes. Ce langage met l'accent sur la description de la structure de graphe et des propriétés que possèdent les noeuds et les relations.

Ce chapitre présente le langage de requête CMQL(Cognitive Map Query Language) pour interroger le modèle des cartes cognitives ontologiques. CMQL est un langage déclaratif qui se base sur les primitives d'accès présentées au chapitre précédent. Les primitives permettent de structurer les différents objets du modèle et d'y accéder sous forme de relations. La syn-

taxe générale de CMQL est proche de celle de SQL ou SPARQL, elle utilise les clauses SELECT-FROM-WHERE. Les formules primitives sont des formules atomiques du langage, elles peuvent être combinées par les opérateurs classiques de la logique du premier ordre. La sémantique des formules, dont les formules primitives, est basée sur celle du calcul relationnel de domaine. Chaque formule dénote une relation dont les attributs sont des objets du modèle.

Ce chapitre est composé de 4 parties. La première partie est bibliographique, elle présente les langages de requête SPARQL et Cypher destinés respectivement aux modèles de graphe RDF et aux graphes de propriétés. La deuxième partie présente la syntaxe du langage de requête CMQL. La troisième partie présente un exemple de requête CMQL et introduit les notions de variables libres et liées. La quatrième partie présente la sémantique de CMQL par une définition dénotationnelle puis explique la sémantique d'une requête au travers d'un exemple.

4.1 Présentation de SPARQL et Cypher

Cette partie présente deux langages de requêtes dans le but de donner un bref aperçu des langages d'interrogation des modèles graphiques de connaissances. Nous présentons d'abord SPARQL qui est un langage de requête particulièrement utilisé dans le domaine de la représentation des connaissances, il permet d'interroger le modèle RDF. Nous présentons ensuite Cypher, un langage qui permet d'interroger des bases de données de graphes de type NoSql qui sont basées sur le modèle des graphes de propriétés.

Description de SPARQL

SPARQL [PS06] (Sparql Protocol And RDF Query Language) est une technologie du web sémantique et une recommandation W3C qui regroupe un protocole pour RDF [CFT08], un langage de requête pour RDF et un format XML pour les résultats des requêtes. Nous nous concentrons ici sur le langage de requête destiné à l'interrogation de connaissances RDF. Tout comme la structure de ces connaissances, le langage SPARQL est basé sur la description de triplets RDF et de graphes composés de ces triplets. Ce langage propose quatre types de requêtes différents [PS06] :

— Les requêtes « ASK » sont les plus simples, elles permettent de vérifier l'existence d'un graphe RDF vérifiant des contraintes en retournant un booléen.
— Les requêtes « DESCRIBE » permettent d'obtenir des informations sur les résultats de la requête vérifiant les contraintes spécifiées.
— Les requêtes « SELECT », qui sont les plus utilisées [PV11], permettent d'extraire des connaissances vérifiant les contraintes spécifiées.

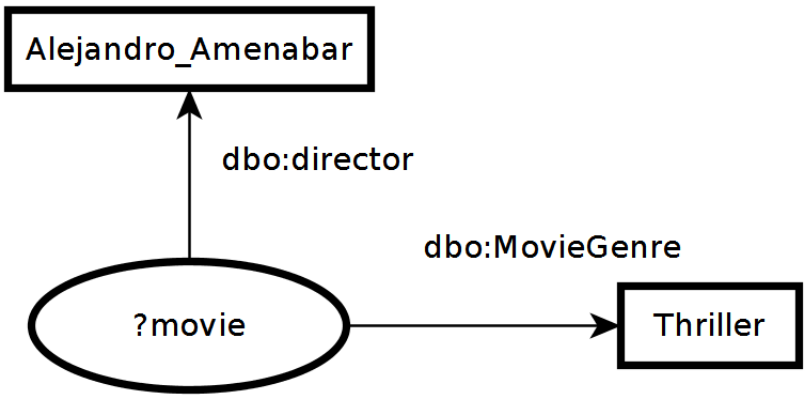


FIGURE 4.1 – Exemple patron de graphe

— Les requêtes « CONSTRUCT » sont similaires aux requêtes « SELECT » mais retournent les résultats sous forme de graphes RDF.

Bien que ces types de requêtes soient différents, ils permettent tous de spécifier des contraintes de la même manière. Ces contraintes sont spécifiées dans une clause « WHERE » par des patrons de triplets ou de graphes qui peuvent être combinés par disjonction, conjonction ainsi que d'autres opérateurs.

Un *triplet* RDF est un triplet (Sujet, Predicat, Objet), c'est la structure de base des connaissances RDF. Le sujet est une ressource à décrire, le prédicat une propriété de cette ressource et l'objet, qui peut être une autre ressource, est la valeur de cette propriété pour la ressource décrite. Un patron de triplet est l'élément atomique pour la description de contraintes dans les requêtes SPARQL. Un patron de triplet est un triplet dont les éléments sont soit des variables soit des constantes. Les variables sont exprimées syntaxiquement par leur nom préfixé du symbole « ? ». Les patrons de triplets permettent d'identifier les triplets qui satisfont les contraintes décrites par les constantes du patron.

Un patron de graphe est un ensemble de patrons de triplets qui sont connectés entre eux par les sujets ou les objets qu'ils ont en commun. Les variables présentes dans plusieurs patrons de triplets sont appelées *variables de jointures*, les patrons de graphes permettent ainsi d'exprimer plusieurs contraintes sur les variables de jointures en utilisant plusieurs patrons de triplets. Les patrons de graphe permettent d'identifier des sous-graphes RDF qui satisfont les contraintes décrites.

Exemple 56. *Prenons par exemple le patron de graphe de la figure 4.1. Ce patron de graphe est composé de deux patrons de triplets :*

— *?movie dbo :Director 'Alejandro_Amenabar'*[1]

1. « dbo » est le préfixe utilisé pour l'ontologie DBpedia dont l'exemple est inspiré. 'Alejandro_Amenabar' est ici

— *?movie dbo :MovieGenre 'Thriller'*

Ces deux patrons de triplets partagent la même variable de jointure ?movie *et contraignent sa valeur par les contraintes exprimées par des constantes. Il est cependant possible qu'un patron de graphe soit composé de plusieurs sous-graphes qui ne sont pas liés par des variables de jointures.*

Les patrons de triplets et de graphes sont la base des spécifications d'une requête SPARQL, mais ce langage propose de nombreux opérateurs, certains similaires à ceux de SQL comme ORDERBY ou DISTINCT, et d'autres propres à SPARQL comme OPTIONAL. Nous ne les présentons pas ici et invitons le lecteur à se référer à [PAG09].

Syntaxe de SPARQL

La syntaxe des requêtes SPARQL est définie exhaustivement dans le cadre de la recommandation W3C [Pru08]. Nous présentons ici la forme syntaxique des requêtes SPARQL d'une manière très générale. La syntaxe d'une requête suit les règles suivantes :

Q ::=	Prologue (SelectQ | ConstructQ | DescribeQ | AskQ)
SelectQ ::=	'SELECT' (VarMode) ? (Var+|'*') DatasetC* WhereC ModifC
ConstructQ ::=	'CONSTRUCT' ConstructTemplate DatasetC* WhereC ModifC
DescribeQ ::=	'DESCRIBE' (VarOrIRIref+ | '*') DatasetC* WhereC ? ModifC
AskQ ::=	'ASK' DatasetC* WhereC
DataSetC ::=	'FROM' ('NAMED') ? GraphNameC
WhereC ::=	'WHERE' ? '{' TriplesC ? ((OperationPattern) '.' ? TriplesC ?)* '}'
ModifC ::=	OrderClause ? LimitOffsetClauses ?

Une requête SPARQL est composée d'un prologue et d'un des quatre types différents de requêtes. Le prologue contient un ensemble de préfixes qui sont des abréviations d'IRI pouvant être utilisées dans la requête, cet ensemble peut être vide. Un mot-clé permet ensuite d'identifier le type de requête, comme par exemple « SELECT » ou « ASK ». Les DataSetClause, WhereClause et ModifierClause sont communes à différents types de requêtes, bien que la WhereClause soit optionnelle pour les requêtes de type *Describe* et que la ModifierClause ne soit pas présente dans les requêtes de type *Ask*. La DataSetClause permet d'indiquer les graphes RDF à interroger en utilisant le mot-clé « FROM », cette clause est cependant optionnelle pour tous les types de requêtes, une source de donnée par défaut peut en effet être définie par le système. La WhereClause permet d'exprimer les contraintes en utilisant les patrons de graphes et divers opérateurs comme « FILTER » ou « UNION », elle est composée du mot-clé « WHERE »(optionnel) suivie d'accolades comprenant la description des patrons

présenté comme un littéral à des fins de simplicité, on trouverait généralement une ressource de type Person ou Director.

et des opérateurs. La ModifierClause sert à effectuer des manipulations sur le résultat de la requête, par exemple des tris ou limitations dans les résultats. Les différents types de requêtes peuvent également avoir des clauses spécifiques comme les requêtes *Select* qui permettent d'indiquer par exemple si les résultats doivent être distincts en utilisant la clause *VarMode* ou les requêtes *Construct* qui permettent de définir un modèle de construction du graphe résultat.

Exemple 57. *Cet exemple illustre une requête SPARQL bien formée de type* Select *:*

```
PREFIX dbo:<http://dbpedia.org/ontology/>
SELECT DISTINCT ?name WHERE {
?filmURI a dbo:Film .
?filmURI dbo:FilmRuntime ?duration .
?filmURI dbo:FilmTitle ?name
} ORDER BY ?name DESC(?duration) LIMIT 10
```

Elle permet d'obtenir le nom des 10 films les plus longs sur DBpedia.

Sémantique de SPARQL

L'interprétation et l'évaluation des requêtes SPARQL se base sur une sémantique qui associe des actions à des mots-clés [Pel16]. La sémantique complète de SPARQL ainsi que la complexité d'évaluation des requêtes sont définies exhaustivement par Pérez et al [PAG09], nous donnons ici une idée générale du sens et de l'évaluation des patrons de triplets et des patrons de graphes.

Formellement, le sens d'un patron de triplet est défini comme étant un élément de l'ensemble $(U \cup V) \times (U \cup V) \times (U \cup L \cup V)$ où U, L et V sont respectivement les ensembles des URIs d'un graphe RDF, l'ensemble des littéraux d'un graphe RDF et l'ensemble des variables. Le sens d'un patron de graphe est l'intersection des patrons de triplets qui le composent, l'ensemble des variables d'un patron de graphe est l'union des variables des patrons de triplets qui le composent.

L'évaluation d'un patron de graphe, appelée *application*, est le calcul des valeurs des variables de ce patron de graphe telles qu'en remplaçant chaque variable par sa valeur dans le patron de graphe, on obtient un sous-graphe du graphe RDF interrogé. Formellement une application se définit par une fonction des variables V d'un patron de triplet vers les ressources du graphe RDF (URI ou littéraux), ainsi à chaque patron de triplet est associé un ensemble d'applications qui font correspondre les variables à leur valeur. Les patrons de graphes étant des intersections de patrons de triplets, un opérateur est utilisé pour faire la jointure entre les applications des différents patrons de triplets, cet opérateur fait l'union des applications tels qu'elles sont compatibles.

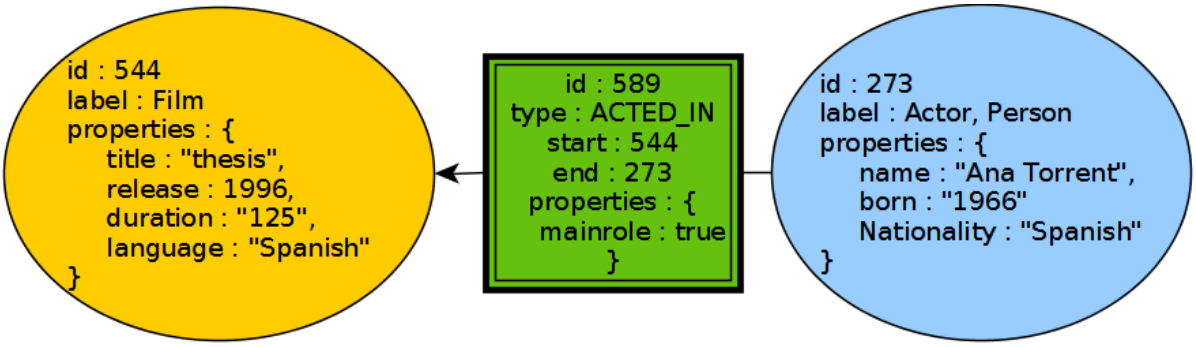


FIGURE 4.2 – Extrait de graphe de propriétés.

Description de Cypher

Les bases de données orientées graphe utilisent pour la plupart le modèle de graphe de propriétés. Ces bases de données sont particulièrement utilisées dans l'industrie mais également dans des travaux de recherche [LMV16], différents langages de requête [Res+16 ; Rod15] ont été développés pour ces nouvelles bases, comme le langage Cypher [Ang+17 ; Fra+18b ; LMD14], qui est un des plus utilisés. Un projet de standardisation est en cours pour concevoir GQL [Pla19], un langage de requête pour graphes de propriétés, en reprenant les caractéristiques principales des langages Cypher, PGQL [Res+16] et G-CORE [Ang+18]. Nous présentons ici Cypher, ses caractéristiques principales, sa syntaxe et sa sémantique.

Comme la plupart des langages de requêtes, Cypher est inspiré de SQL, mais aussi de SPARQL et de langages de programmation comme Python ou Haskell. Cypher permet d'effectuer des requêtes pour lire les connaissances, pour les modifier et pour administrer la base, nous nous concentrons ici sur l'interrogation des connaissances. Cypher propose différentes clauses à exprimer dans une requête, elles sont placées bout à bout de sorte à ce que le contexte d'une clause soit le résultat de la clause précédente. C'est d'ailleurs pourquoi la clause *Return*, qui permet d'indiquer les variables souhaitées dans le résultat, se situe à la fin de la requête à l'inverse de SQL ou SPARQL. Parmi les clauses les plus importantes on peut citer celles-ci :

— *Match* : C'est l'équivalent de la clause *Where* en SPARQL, elle permet d'exprimer des contraintes sur la structure du graphe. La clause *OptionalMatch* est similaire mais permet de retourner certains éléments quand ils sont disponibles, « null » sinon, comme *Optional* en SPARQL.

— *With* : Cette clause permet d'intervenir sur le chaînage des clauses en modifiant des valeurs entre deux clauses ou en introduisant des agrégats. On peut par exemple trouver dans cette clause, les clauses *Limit* ou *OrderBy*.

— *Where* : Cette clause permet d'ajouter des contraintes aux clauses *Match*, *OptionalMatch* et *With*. Elle pourrait être vue comme l'équivalent d'un *Filter* en SPARQL mais elle est

beaucoup plus expressive, permettant par exemple des requêtes imbriquées.

— *Return* : Cette clause permet d'indiquer le résultat souhaité.

— *Union* : Cette clause permet de combiner les résultats de plusieurs requêtes.

— ...

Cypher est un langage qui met l'accent sur la description des contraintes structurelles du graphe, appelés *motifs*. La syntaxe choisie pour la description des motifs est très visuelle et expressive, permettant par exemple de spécifier des relations avec des flèches (« (a)-[rel]->(b) ») et de décrire des chemins dans le graphe de différentes façons.

Syntaxe de Cypher

La syntaxe des requêtes Cypher est définie exhaustivement dans la littérature [Fra+18a ; Fra+18b] ainsi que sur le site de Neo4j [2]. Nous présentons ici la forme syntaxique générale des requêtes Cypher ainsi que la syntaxe des expressions de motifs. La syntaxe générale d'une requête suit les règles suivantes :

Q ::=	subQ | Q 'UNION' ('ALL') ? Q
subQ ::=	'RETURN' res | clause subQ
res ::=	'*' | expr ('AS' a) ? | res ',' expr ('AS' a) ?
clause ::=	('OPTIONAL') ? 'MATCH' patterns ('WHERE' expr) ?
	| 'WITH' res ('WHERE' expr) ?
	| 'UNWIND' expr 'AS' a
patterns ::=	pattern | pattern ',' patterns

Une requête (Q) est une sous-requête (subQ) ou une combinaison de sous-requêtes. Le mot-clé « ALL », pouvant être utilisé lors de combinaisons de requêtes, sert à indiquer que l'on souhaite garder les doublons qui sont autrement supprimés. Une sous-requête est un enchaînement de clauses se terminant par la clause *Return* qui sert à indiquer le résultat (res) souhaité par un enchaînement d'expressions (expr) pouvant utiliser des alias (avec le mot-clé « AS » comme en SQL). Différentes clauses peuvent s'enchaîner, comme *Unwind* qui sert à manipuler des listes ou *Match* qui permet de spécifier les contraintes structurelles de matching en utilisant des motifs.

Exemple 58. *Cet exemple illustre une requête Cypher bien formée :*

```
MATCH (m:Film {title:'Thesis'})<-[:ACTED_IN]-(a:Actor)-[:ACTED_IN]->(m2:Film)
WHERE m2.language = "Spanish"
RETURN a.name, m2.title
```

Elle permet de savoir dans quels films en espagnol les acteurs du film Thesis *ont joué.*

2. https://neo4j.com/docs/cypher-manual/current

Comme on peut le voir dans la clause *match* de l'exemple précédent, les motifs constituent une partie importante des requêtes Cypher. En voici la syntaxe générale :

```
    Pattern ::= subPattern | a '=' Pattern
subPattern ::= nodeP | nodeP relP subPattern
     nodeP ::= '(' a ? labels ? map ? ')'
      relP ::= '-[' a ? types ? length ? map ? ']-'
             | '<-[' a ? types ? length ? map ? ']-'
             | '-[' a ? types ? length ? map ? ']->'
    labels ::= ' :' l | ' :' l labels
     types ::= ' :' t | ' :' t types
    length ::= '*' | '*'n | '*'n'..' | '*..'n | '*'n'..'n|
```

Un motif est soit assigné à une variable, notamment pour créer des variables de chemins que nous détaillerons en expliquant la clause *Length*, soit exprimé directement. Les motifs contiennent des expressions de noeud (nodeP) déclarées entre parenthèses et des expressions de relations (relP) déclarées entre crochets. Il est possible de former des chemins en liant plusieurs noeuds avec des relations comme illustré dans l'exemple précédent où le noeud « (a :Actor) » est mis en relation avec deux noeuds représentant des films. Les noeuds et les relations peuvent être caractérisés par des variables (a), leurs types (types), et une collection d'attributs placés entre accolades (map). Les chemins peuvent être décrits directement dans une relation grâce à la clause *Length* qui permet de déterminer la longueur d'une chaîne de relations.

Exemple 59. *Cet exemple illustre la syntaxe des chemins en utilisant la clause* Length*.*

- `(a)-[p*]->(b)` *décrit tous les chemins p de a vers b quelles que soit leur taille.*
- `(a)-[p*3]-(b)` *décrit les chemins p de longueur 3 entre a et b.*
- `(a)-[*..2]-(b)` *décrit les chemins de longueur inférieure ou égale à 2 entre a et b.*
- `(a)-[p:KNOWS*2..3]->(b)` *décrit les chemins p de longueur 2 ou 3 de a vers b dont les relations sont de type* knows*.*

Sémantique de Cypher

Le langage Cypher est très expressif, contenant beaucoup de clauses différentes, cela rend la description de sa sémantique complexe. C'est pourquoi nous ne la définissons pas ici en détail, nous en présentons l'idée générale, la sémantique formelle et détaillée est cependant disponible dans la littérature [Fra+18a ; Fra+18b].

Les éléments clés de Cypher pour définir sa sémantique sont les suivants [Fra+18a] :

- Un modèle de données comprenant *valeurs*, *graphes* et *tables*.

— Un langage comprenant des *expressions*, des *motifs*, des *clauses* et des *requêtes*.

La sémantique de Cypher se base sur la définition d'une relation et deux fonctions :

— La relation de *pattern matching*(correspondance de motifs) vérifie si un chemin p d'un graphe G correspond à un motif π avec les valeurs u assignées aux variables libres du motif, elle est notée : (p,G,u) $\models \pi$.

— La sémantique des *expressions* est une fonction qui associe à une expression e, un graphe G et des valeurs u, une valeur notée $[\![e]\!]_{G,u}$.

— La sémantique d'une *requête* Q (resp. clause C) sur un graphe G est une fonction notée $[\![Q]\!]_G$ (resp. $[\![C]\!]_G$) dont le domaine et codomaine sont des tables.

Conclusion

La plupart des langages de requêtes permettant d'interroger des modèles graphiques s'inspirent de SQL puis proposent une manière déclarative d'exprimer les connaissances désirées. La façon de décrire ces connaissances dépend très fortement de la nature de ces connaissances et donc du modèle de connaissances ciblé. Ainsi, une base RDF étant composée de triplets, SPARQL se base sur l'expression de ces triplets et les graphes de propriétés modélisant des graphes dont les noeuds et les relations peuvent avoir diverses étiquettes et propriétés, Cypher se base sur l'expression des motifs de graphe et de propriétés.

Les cartes cognitives ontologiques représentent des influences directes ou indirectes, des relations taxonomiques mais aussi des valeurs d'influences propagées ou encore des assertions spatio-temporelles. Les spécificités des cartes cognitives rendraient les langages décrits inadaptés à l'interrogation du modèle. La variété des objets du modèle demande une certaine structure, c'est pour cela que les primitives ont été conçues : afin de mettre en relation les différents objets du modèle. Les prochaines parties présentent donc le langage CMQL, qui se base sur les primitives d'accès, et s'inspire des langages existants comme SQL, DRC ou SPARQL.

4.2 Syntaxe de CMQL

La syntaxe de CMQL est intuitive, elle s'inspire de celles de SQL et de SPARQL en utilisant les clauses *Select*, *From*, *Where* et *Modifiers*. Ces clauses permettent respectivement de sélectionner les variables désirées comme résultat, sélectionner la source des données, spécifier les contraintes sur les variables du résultat et appliquer des modifications au résultat. De manière générale une *requête* est principalement composée de *variables* à retourner, d'une source de donnée et de *formules* qui servent à spécifier des contraintes sur les variables. Ces formules sont définies récursivement et combinables entre elles selon différentes règles. Les

formules atomiques sont soit des *formules primitives* soit des *formules d'expressions*. Les formules primitives, présentées au chapitre précédent, sont composées d'un nom de primitive et de *termes* étant soit des variables soit des *constantes*. Les formules d'expressions sont composées de deux termes et d'un *opérateur*. Nous présentons ici une syntaxe détaillée du langage CMQL sous la forme de Backus-Naur (BNF) [Knu64 ; MR03] accompagnée d'explications avant de fournir un exemple illustrant un arbre syntaxique.

```
Query ::=  SelectClause FromClause WhereClause ModifierClause ?
```

La forme générale d'une requête CMQL est composée de trois, voire quatre, clauses : une clause *Select*, une clause *From*, une clause *Where* et éventuellement une clause *Modifier*. Cette forme générale est très courante dans les langages de requêtes [Mih96 ; MZ97 ; PS06 ; Res+16].

```
SelectClause ::=  'SELECT' ResultsClause (',' ResultClause)*
ResultClause ::=  ResultMode '(' 'DISTINCT' ? Var ')' | 'DISTINCT' ? Var
  ResultMode ::=  'COUNT' | 'AVG' | 'SUM' | 'MIN' | 'MAX'
```

La clause *Select* est toujours débutée par le mot-clé « SELECT » puis est composée d'une ou plusieurs clauses de résultats séparés par des virgules. Une clause de résultat est composée d'une variable éventuellement précédée du mot-clé « DISTINCT ». Une clause de résultat peut aussi être composée d'une clause indiquant une fonction d'agrégation du résultat, auquel cas un mot-clé (« COUNT »,« SUM »...) est attendu et le reste de la clause de résultat sera compris entre parenthèses.

```
FromClause ::=  'FROM' ('ALL' | MapName ( ',' MapName)*)
```

La clause *From* est toujours débutée par le mot-clé « FROM » puis est composée soit du mot-clé « ALL » soit d'un ou plusieurs noms de cartes séparés par des virgules.

```
WhereClause ::=  'WHERE' '{' FormulaClause '}'
```

La clause *Where* est toujours débutée par le mot-clé « WHERE » puis composée d'une accolade ouvrante, d'une clause *Formula* et d'une accolade fermante.

```
ModifierClause ::=  ('GROUPBY' Var (',' Var)) ?
                    (ORDERBY resultClause (',' resultClause)) ?
                    (LIMIT Const) ?
                    (OFFSET Const) ?
```

La clause *Modifier* est composée de quatre opérations optionnelles : l'opération débutant par le mot-clé « GROUPBY » suivie d'une ou plusieurs variables séparées par des virgules, l'opération débutant par « ORDERBY » suivie d'une ou plusieurs clauses de résultat séparées par des virgules et les opérations débutant respectivement par les mot-clés « LIMIT » et « OFFSET » qui sont suivies d'une constante désignant un entier naturel supérieur ou égal à 0.

```
FormulaClause ::= 'NOT' '(' formulaClause ')'
                | '(' formulaClause ')'
                | FormulaClause 'AND' FormulaClause
                | FormulaClause 'OR' FormulaClause
                | 'ALL' var (',' var)* '(' FormulaClause ')' '->' '(' FormulaClause ')'
                | 'ALL' var (',' var)* '(' FormulaClause ')'
                | 'SOME' var (',' var)* '(' FormulaClause ')'
                | AtomicClause
```

La clause *Formula* est une règle récursive ou une formule atomique. La règle de négation est commencée par le mot-clé « NOT » puis suivie d'une clause *Formula* entre parenthèses. La règle de priorité est composée d'une clause *Formula* entre parenthèses. Les règles de conjonction et de disjonction sont composées de deux clauses *Formula* séparées respectivement par les mot-clés « AND » et « OR ». Les règles de quantification universelle et existentielle sont commencées respectivement par les mot-clés « ALL » et « SOME », puis suivies d'une ou plusieurs variables séparées par des virgules puis d'une clause *Formula* entre parenthèses. La règle d'implication est composée d'un début correspondant à la règle de quantification universelle suivie d'une flèche (« -> ») et d'une seconde clause *Formula* entre parenthèse.

```
AtomicClause ::= ExpressionClause | PrimitiveClause
```

Une formule atomique est soit une formule d'expression, soit une formule primitive.

```
ExpressionClause ::= T Operator T
        Operator ::= '<' | '<=' | '=' | '!=' | '>=' | '>'
               T ::= Var | Const
             Var ::= '?' VarName
```

Une formule d'expression est composée d'un terme, d'un opérateur et d'un second terme. Les opérateurs sont les opérateurs binaires classiques de comparaison (« < », « = »...). Un terme est soit une constante, soit une variable. Les variables sont composées du nom de la variable et préfixées par le symbole « ? ».

```
PrimitiveClause ::= PrimitiveName '(' T (',' T)* ')'
```

Une formule primitive est composée du nom de la primitive, d'une parenthèse ouvrante, de termes séparés par des virgules et d'une parenthèse fermante.

La syntaxe présentée ne permet pas, à elle seule, de spécifier la validité d'une requête, plusieurs règles de syntaxe supplémentaires doivent en effet être définies afin de contraindre le langage. Six règles sont présentées ci-dessous :

1. Les variables résultats de la clause *Select* sont les variables libres de la requête, elles ne peuvent donc pas être quantifiées. Ces variables doivent être distinctes les unes des autres et toutes présentes dans la formule de la clause *Where*.
2. Dans les règles de quantification universelle et existentielle, les variables var doivent être distinctes les unes des autres et libres dans F.
3. Dans la règle d'implication, les variables var doivent être distinctes les unes de autres et libres dans F1 ou F2 (ou les deux).
4. Les variables étant associées à des domaines dans les primitives, une variable présente à plusieurs reprises dans une requête doit toujours être associée au même domaine.
5. Pour une formule d'expression, les termes (variables et constantes) utilisés doivent être du même domaine, et l'opérateur utilisé doit par ailleurs être défini pour ce domaine.
6. Au moins une formule primitive doit être présente dans la formule de la clause *Where*, les variables d'une formule d'expression doivent apparaître dans au moins une formule primitive, il doit par ailleurs y avoir au moins une variable dans une formule d'expression.

4.3 Un exemple de requête CMQL

Cette partie décrit un exemple de requête CMQL et en détaille l'arbre syntaxique. Cette requête s'applique à l'ensemble de cartes cognitives ontologiques S_1 qui comprend les cartes cognitives M1 et M2 (figure 4.4) définies sur l'ensemble de valeurs I={-4,-3,-2,-1,0,1,2,3,4}, la taxonomie T1 (figure 4.3) et l'ontologie spatio-temporelle OST1 qui n'est pas présentée ici.

Exemple 60. *La requête CMQL $Q1$ est l'exemple utilisé dans cette partie :*

```
 SELECT ?c2 FROM ALL WHERE{
Path(?m,?c1,?c2,?p) AND KindOf(?c1, Reglement)
}
```

Deux primitives sont utilisées dans cette requête : la primitive Path est une relation entre une carte, deux concepts et un chemin telle que les deux concepts sont le concept source et

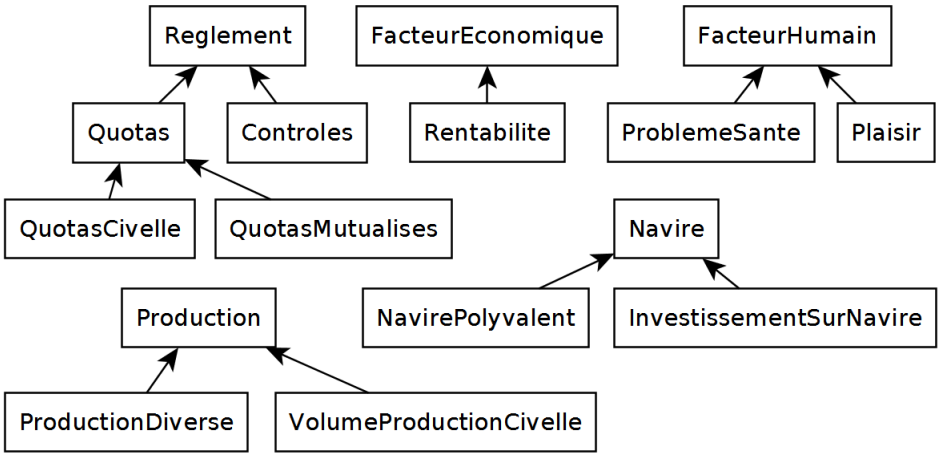


FIGURE 4.3 – Taxonomie de concepts T1.

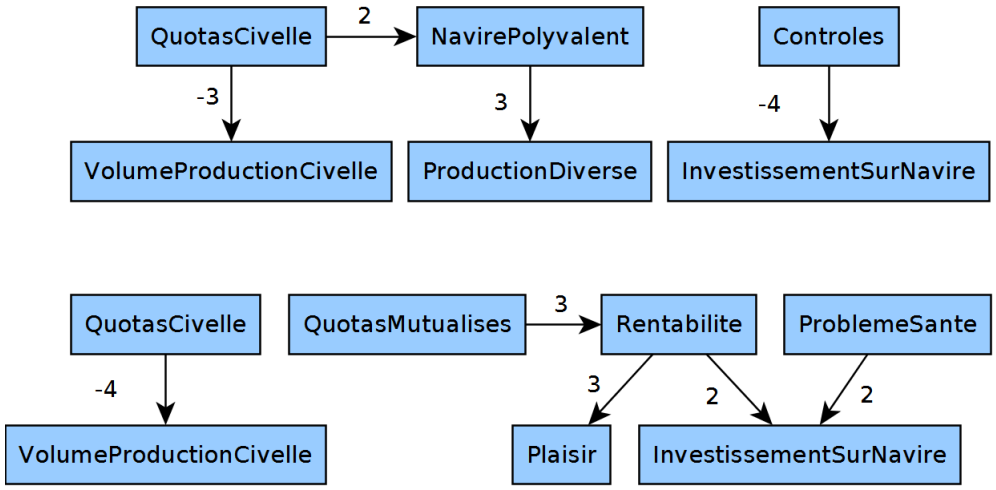


FIGURE 4.4 – Cartes cognitives ontologiques M1 (haut) et M2 (bas).

le concept destination du chemin dans la carte, la primitive KindOf est une relation entre deux concepts telle que le premier concept est une spécialisation du second dans la taxonomie. Cette requête sélectionne les concepts tels qu'il existe un chemin dans une carte menant à ce concept et dont le concept source est une spécialisation du concept « Règlement ». Formulé plus simplement, cette requête permet de sélectionner les concepts qui sont influencés par une sorte de « Règlement ».

On remarque que la requête de l'exemple précédent ne respecte pas la règle de grammaire stipulant que les variables résultats de la requête sont celles qui sont libres dans la requête, toutes les variables à part ?c2 doivent en effet être quantifiées dans la requête.

Cette requête est cependant valide, car en l'absence de quantification explicite, les variables sont considérées comme quantifiées existentiellement. Lorsqu'une variable n'est pas explicite-

ment quantifiée, une reformulation de requête est effectuée. Le quantificateur « SOME » est alors placé à la branche de l'arbre syntaxique la plus profonde contenant toutes les occurrences de la variable à quantifier. Une fois reformulée la requête de l'exemple précédent devient la requête $Q1'$:

```
SELECT ?c2 FROM ALL WHERE {
SOME ?c1 (SOME ?m,?p (Path(?m,?c1,?c2,?p)) AND KindOf(?c1,Règlement))
}
```

La figure 4.5 ci-dessous représente une version simplifiée de l'arbre syntaxique de cette requête.

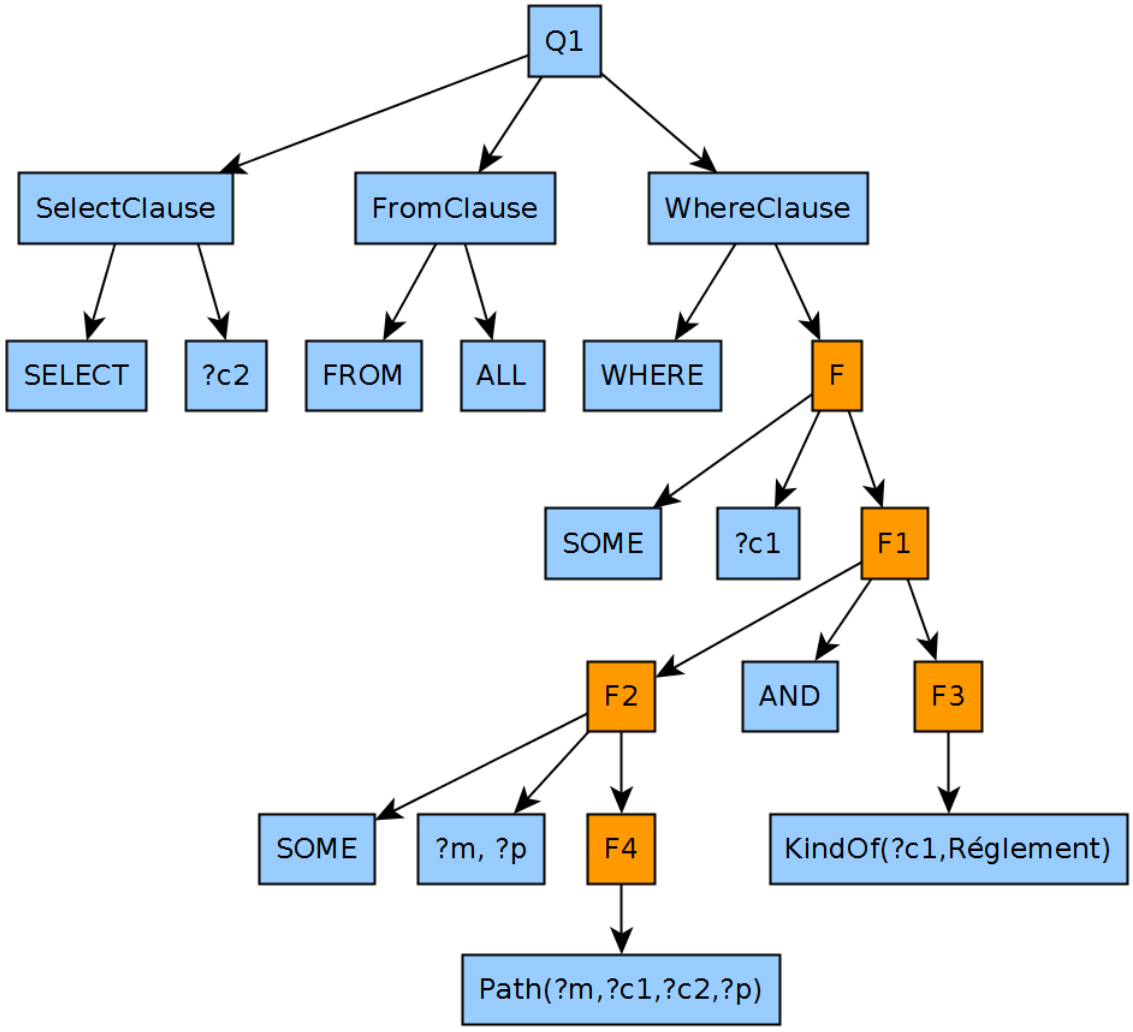


FIGURE 4.5 – Arbre syntaxique simplifié de la requête $Q1'$.

Les formules sont identifiées par les noeuds en orange, ainsi F est la formule `SOME ?c1 ( F1 )`, F1 est la formule `F2 AND F3`, etc... Lors de la reformulation de la requête, la formule F2 a été insérée à la place de F4 pour quantifier les variables ?m et ?p qui n'apparaissent que dans

F4. La formule F a été insérée à la place de F1 pour quantifier ?c1 car F1 est la formule la plus profonde de l'arbre qui contient toutes les occurrences de ?c1, apparaissant à la fois dans F2 et F3. Toutes les variables (à part ?c2) sont alors liées dans la formule F, et donc dans la requête. Une variable n'est pas intrinsèquement libre ou liée, elle l'est par rapport à un contexte, une formule. Dire qu'une variable est une variable libre de la requête signifie qu'elle est libre dans le contexte de la formule la plus générale de la requête (ici F). Le tableau ci dessous illustre l'état, libre ou lié, des variables dans le contexte des différentes formules. L signifie Libre, Q signifie Quantifiée (liée) et X signifie que la variable n'existe pas dans le contexte de cette formule.

	F	F1	F2	F3	F4
?c2	L	L	L	X	L
?c1	Q	L	L	L	L
?m	Q	Q	Q	X	L
?p	Q	Q	Q	X	L

Chaque formule est indépendante de son contexte, elle aura donc la même structure et la même valeur où qu'elle soit placée dans la requête, on dit que la sémantique des formules est strictement « bottom-up » (de bas en haut). Toutes les variables sont donc libres dans F4, *?c2*, *?m*, *?p* n'existent pas dans F3. *?m* et *?p* ne sont pas libres dans F, F1 et F2 étant quantifiées dans F2, *?c1* n'est pas libre dans F étant quantifiée dans F.

4.4 Sémantique de CMQL

Cette partie présente la sémantique du langage CMQL, pour cela une définition dénotationnelle [All86 ; Mos90 ; Ten76] est proposée. Une telle méthode, basée sur la théorie des domaines [Sco82], vise à définir le sens d'un langage en définissant des fonctions des structures syntaxiques vers des objets mathématiques. La syntaxe abstraite du langage est définie afin d'introduire les différentes composants et structures syntaxiques utilisés lors de la définition dénotationnelle de la sémantique.

Les composants syntaxiques sont les suivants : Q (Requête), M (Nom de carte cognitive), F (Formule), P(Formule primitive), E (Expression comme $x < 3, y = z$ etc...), N (Nom de primitive), T (Terme), V (Variable), C (Constante) et OP (Opérateurs comme $=, >, <$...). La structure syntaxique est présentée au format BNF dans la syntaxe abstraite suivante :

$\langle Q \rangle$ ::= 'SELECT' $\langle V \rangle$+ 'FROM' $\langle M \rangle$+ 'WHERE' $\langle F \rangle$

$\langle F \rangle$ ::= 'NOT'$\langle F \rangle$
| '('$\langle F \rangle$')'

| ⟨*F*⟩ 'AND' ⟨*F*⟩
| ⟨*F*⟩ 'OR' ⟨*F*⟩
| 'SOME' ⟨*V*⟩+ '('⟨*F*⟩')'
| 'ALL' ⟨*V*⟩+ '('⟨*F*⟩')'
| 'ALL' ⟨*V*⟩+ '('⟨*F*⟩')' '(' ⟨*F*⟩ ')'
| ⟨*P*⟩
| ⟨*E*⟩

⟨*P*⟩ ::= ⟨*N*⟩ '('⟨*T*⟩+')'

⟨*E*⟩ ::= ⟨*T*⟩ ⟨*OP*⟩ ⟨*T*⟩

⟨*T*⟩ ::= ⟨*V*⟩
| ⟨*C*⟩

L'idée principale derrière la définition de la sémantique de CMQL est que chaque requête, tout comme chaque formule, dénote une relation dont les attributs sont les variables libres de la requête ou de la formule.

Définition 48 (Requête CMQL)**.** *Soit S un ensemble de cartes cognitives ontologiques définies sur la taxonomie T, l'ensemble de valeurs I et l'ontologie O_{ST}. Le sens d'une requête Q est une fonction d'un sous-ensemble de S vers une relation dont les attributs sont les variables libres de la requête : $mng_q : 2^S \rightarrow 2^{\prod(V:D)}$ où V est l'ensemble des variables libres avec D l'ensemble de leurs domaines respectifs, et $2^{\prod(V:D)}$ (resp. 2^S) l'ensemble puissance de $\prod(V : D)$ (resp. S) c'est-à-dire l'ensemble de tous les sous-ensembles de $\prod(V : D)$ (resp. S).*

Exemple 61. *Prenons l'exemple de la requête Q1 : son sens est une fonction mng_{Q1} qui associe à l'ensemble des cartes S(du fait du mot-clé « ALL »), un sous-ensemble du produit cartésien des domaines des variables libres de la requête. Etant donné qu'il n'y a qu'une variable libre $?c2$ dont le domaine est l'ensemble des concepts de la taxonomie C, le résultat sera un sous-ensemble de C. Si la variable $?m$, dont le domaine est l'ensemble des cartes S, était également une variable libre de la requête, le résultat de la requête aurait été un sous-ensemble du produit cartésien de S et de C.*

D'une manière générale, le sens d'une formule est une relation dont les attributs sont les variables libres de la formule.

Définition 49 (Formule CMQL)**.** *Le sens d'une formule CMQL est une fonction $mng_f : F \rightarrow \prod(V : D)$ où V est l'ensemble des variables libres de la formule et D l'ensemble de leurs domaines respectifs. Le sens plus spécifique de chaque règle de formule de la syntaxe abstraite est présenté ci-dessous.*

Soit $fr(F)$ l'ensemble des variables libres de la formule F et $rel[V_i, V_j]$ la relation rel tronquée ayant deux attributs V_i, V_j.

1. $mng_f(\ NOT\ F_1\ ) = \prod(fr(F_1) : D) - mng_f(F_1)$.

2. $mng_f(\ (F)\ ) = mng_f(F)$

3. $mng_f(\ F_1\ AND\ F_2\ ) =$
   $mng_f(F_1) \times \prod(fr(F) - fr(F_1) : D) \bigcap$
   $mng_f(F_2) \times \prod(fr(F) - fr(F_2) : D)$

4. $mng_f(\ F_1\ OR\ F_2\ ) =$
   $mng_f(F_1) \times \prod(fr(F) - fr(F_1) : D) \bigcup$
   $mng_f(F_2) \times \prod(fr(F) - fr(F_2) : D)$

5. $mng_f(\ SOME\ V_1..V_n\ F_1\ ) = mng_f(F_1)[fr(F_1) - \{V_1, ..., V_n\}]$

6. $mng_f(\ ALL\ V_1..V_n\ F_1\ ) =$
   $\{t \in \prod(fr(F_1) - \{V_1, ..., V_n\} : D)\ /\ \{t\} \times \prod(\{V_1, ..., V_n\} : D) \subseteq mng_f(F_1)\}$

7. $mng_f(\ ALL\ V_1..V_n\ F_1\ \rightarrow\ F_2\ ) =$
   $\{t \in \prod(\ (fr(F_1) \bigcup fr(F_2)) - \{V_1..V_n\} : D)\ /$
   $\{t_1[\{V_1..V_n\}]\ / t_1 \in mng_f(F_1)\ et\ t_1[fr(F_1) - \{V_1..V_n\}] = t[fr(F_1) - \{V_1..V_n\}]\}$
   $\subseteq \{t_2[\{V_1..V_n\}]\ / t_2 \in mng_f(F_2)\ et\ t_2[fr(F_2) - \{V_1..V_n\}] = t[fr(F_2) - \{V_1..V_n\}]\}\}$

8. $mng_f(\ P(A_1 : T_1, ..., A_n : T_n)\ ) =$
   $\{\ t \in \prod(V : D)\ /\ \exists t_1 \in P,\ \forall i \in [1, n],$
   $t_1(A_i) = T_i\ si\ T_i\ est\ une\ constante,$
   $t_1(A_i) = t(T_i)\ si\ T_i\ est\ une\ variable\ \}$

9. $mng_f(V_1\ OP\ C) = \{t \in \prod(V_1 : D)\ /\ t(V_1)\ OP\ C\}$
   $mng_f(C\ OP\ V_1) = \{t \in \prod(V_1 : D)\ /\ C\ OP\ t(V_1)\}$
   $mng_f(V_1\ OP\ V_2) = \{t \in \prod(\{V_1, V_2\} : D)\ /\ t(V_1)\ OP\ t(V_2)\}$

La règle 1 définit la sémantique de la négation comme une relation dont les attributs sont les variables libres de la formule F1 et la valeur est le produit cartésien indexé des domaines

moins la valeur de F1.
La règle 2 permet de définir des priorités dans l'arbre syntaxique, son sens est trivial.
Les règles 3 et 4 définissent respectivement la sémantique de la conjonction et de la disjonction de formules, elles sont définies de manières similaires. Dans les deux cas, $mng_f(F_1)$ et $mng_f(F_2)$ sont des relations qui peuvent avoir des structures différentes, F1 et F2 n'ayant pas forcément les mêmes variables libres. La conjonction et la disjonction ne s'effectuent pas directement sur $mng_f(F_1)$ et $mng_f(F_2)$ auquel cas $mng_f(F)$ serait l'ensemble vide pour la conjonction ou un ensemble de tuples à structures différentes pour la disjonction dès lors que $fr(F_1) \neq fr(F_2)$. Ces opérations s'effectuent alors sur la relation $mng_f(F_1)$ à laquelle on ajoute les variables qui sont libres dans F2 sans l'être dans F1 avec l'ensemble de leurs domaines et la relation $mng_f(F_2)$ modifiée de manière similaire.
La règle 5, qui définit la sémantique de la quantification existentielle, restreint la structure de la relation $mng_f(F1)$ en supprimant les variables quantifiées dans la formule F.
La règle 6, qui définit la sémantique de la quantification universelle, considère l'ensemble des tuples composés des variables libres de F tels qu'ils appartiennent à $mng_f(F1)$ pour toutes les valeurs des variables quantifiées universellement dans F.
La règle 7 est une variante de la règle 6, elle en reprend la forme générale. Cette règle est équivalente à la règle :
$ALL\ V_1..V_n\ (NOT(F_1)\ OR\ F_2\ )$, elle est définie dans CMQL car elle consitue un cas courant d'utilisation du quantificateur universel. Les variables libres de F sont les variables libres de F1 et les variables libres de F2 sauf celles quantifiées dans F.
La règle 8 est présentée en détail dans le chapitre précédent qui est consacré aux primitives, elle est présente ici dans un but d'exhaustivité des règles sémantiques des formules.
La règle 9 définit la sémantique des formules d'expressions, c'est une relation dont les attributs sont les variables de la formule qui est composée des tuples vérifiant l'expression. La sémantique des différents opérateurs n'est pas définie ici, étant triviale.

Les différentes règles sémantiques présentées permettent désormais d'obtenir la valeur de la requête $Q1'$ introduite précédemment. La requête $Q1'$ contient deux formules atomiques qui sont les formules primitives F3 et F4(figure 4.5). F3 est une formule primitive utilisant la primitive KindOf (voir III.2), elle s'applique à la taxonomie T1(figure 4.3). La valeur de la formule primitive F3 est l'ensemble des 5 tuples (?c1) :

$?c1$
$Reglement$
$Quotas$
$QuotasCivelle$
$QuotasMutualises$
$Controles$

F4 une formule primitive utilisant la primitive Path (voir III.1), elle s'applique aux cartes

cognitives de la figure 4.4. La valeur de la formule primitive F4 est l'ensemble des 12 tuples (?m, ?c1, ?c2, ?p) représentant les 5 chemins de la carte M1 et les 7 chemins de la carte M2. La valeur de la formule F2 (figure 4.5) qui quantifie existentiellement les variables ?m et ?p, est l'ensemble des 12 tuples (?c1, ?c2) provenant de la valeur de la formule F4 :

$?c1$	$?c2$
$QuotasCivelle$	$NavirePolyvalent$
$QuotasCivelle$	$ProductionDiverse$
$NavirePolyvalent$	$ProductionDiverse$
$ProblemeSante$	$InvestissementSurNavire$
...	...

La formule F1 (figure 4.5) est la conjonction des formules F2 et F3. $mng_f(F3)$ contient un attribut $?c1$ et $mng_f(F4)$ contient deux attributs $?c1$ et $?c2$, la conjonction ne peut donc être appliquée directement : elle est effectuée sur $mng_f(F4)$ et $mng_f(F3) \times C$, C étant l'ensemble des concepts de la taxonomie et le domaine de la variable $?c2$. La valeur de la formule F1 est donc l'ensemble des 8 tuples (?c1, ?c2) :

$?c1$	$?c2$
$QuotasCivelle$	$NavirePolyvalent$
$QuotasCivelle$	$ProductionDiverse$
$Controles$	$InvestissementSurNavire$
...	...

La formule F, qui est la formule principale de la requête, quantifie existentiellement la variable ?c1, la valeur de F est alors l'ensemble des 8 tuples :

$?c2$
$VolumeProductionCivelle$
$NavirePolyvalent$
$ProductionDiverse$
$InvestissementSurNavire$
$VolumeProductionCivelle$
$Rentabilite$
$Plaisir$
$InvestissementSurNavire$

La requête $Q1'$ ne contient pas de modificateurs, le résultat de la requête est donc la valeur de la formule F, c'est-à-dire la relation ci-dessus composée des 8 tuples (?c2).

La requête $Q1'$ est le seul exemple de requête proposé dans ce chapitre, pour plus d'exemples le lecteur est invité à lire le chapitre suivant qui contient de nombreux exemples de

requêtes CMQL.

La sémantique des modificateurs de requête classiques, comme `ORDERBY`, `GROUPBY`, `LIMIT` ou `OFFSET`, n'est pas définie en détails dans cette thèse mais peut être trouvée dans la littérature. Le modificateur `GROUPBY` permet d'agréger des tuples selon les valeurs d'une ou plusieurs colonnes, les autres modificateurs considèrent globalement un ordre sur l'ensemble des tuples qui est déterminé par le modificateur `ORDERBY` et effectuent des tris ou des troncatures.

Conclusion

Ce chapitre présente le langage de requête CMQL qui vise à interroger le modèle des cartes cognitives ontologiques. CMQL peut cependant être appliqué au modèle des cartes cognitives taxonomiques ou même au modèle classique des cartes cognitives, auxquels cas une taxonomie par défaut et une ontologie spatio-temporelle vide seraient considérées. Basé sur les primitives d'accès et inspiré des langages SQL et SPARQL pour la syntaxe et du calcul relationnel de domaine pour la sémantique, le langage CMQL permet d'exprimer simplement et de manière déclarative les connaissances désirées. Il présente une syntaxe de type SQL avec les clauses SELECT, FROM et WHERE, rendant son usage intuitif. Sa sémantique, proche de celle du calcul relationnel de domaine, permet de combiner les primitives grâce aux opérateurs de la logique du premier ordre. Plusieurs modificateurs de requête, du type ORDERBY ou LIMIT, sont également proposés comme en SQL ou SPARQL. Le langage CMQL a été implémenté dans le logiciel VSPCC et testé sur un ensemble de cartes cognitives ontologiques provenant d'un cas concret d'utilisation [3], l'interrogation de l'ensemble des connaissances du modèle n'a pas posé de problème particulier.

3. Ces cartes sont les 53 du corpus du projet Kifanlo présenté dans les chapitre 1 et 5.

CHAPITRE 5

EXPLOITATION D'UNE BASE DE CARTES COGNITIVES ONTOLOGIQUES

Introduction

Ce dernier chapitre est consacré au coté applicatif de cette thèse. Les différentes contributions présentées dans les chapitres précédents ont été implémentées au sein du logiciel VSPCC, qui a été développé par Aymeric LeDorze dans le cadre de ses travaux sur les cartes cognitives. Ce logiciel a fourni une base sur laquelle s'appuyer pour implémenter les différentes notions introduites dans cette thèse, que ce soit le modèle de cartes cognitives ontologiques ou le langage CMQL avec l'ensemble des primitives d'accès. Grâce à une collaboration avec des chercheurs en géographie de l'Université de Nantes, nous avons pu accéder au corpus de cartes du projet Kifanlo. Ce corpus réunit une taxonomie de concepts et une cinquantaine de cartes cognitives taxonomiques représentant les stratégies de pêche de différents pêcheurs des Pays de la Loire. Ce corpus a permis de tester les travaux de cette thèse sur des cartes provenant d'un cas concret et réel.

Ce chapitre est composé de deux parties. La première partie présente le corpus de cartes cognitives taxonomiques du projet Kifanlo, et décrit comment l'exploitation de celui-ci peut être facilitée en utilisant CMQL. La seconde partie présente le logiciel VSPCC ainsi que l'implémentation du modèle des cartes cognitives ontologiques et du langage CMQL.

5.1 Exploitation du corpus du projet Kifanlo

Cette partie propose une exploitation des cartes du projet Kifanlo, elle présente le corpus de cartes du projet et la mise en application des contributions de la thèse. Le projet Kifanlo, présenté au chapitre 1, vise à étudier le contexte et l'évolution de l'activité de pêche en région Pays de la Loire. Pour cela des pêcheurs des différents ports et de différentes générations ont été interrogés. Les entretiens ont permis d'élaborer une carte cognitive par pêcheur en se basant sur une taxonomie de concepts.

5.1.1 Présentation du corpus

Le corpus de cartes du projet Kifanlo est composé d'un ensemble de cartes cognitives taxonomiques avec des informations de métadonnées. Nous présentons d'abord les cartes cognitives avant de présenter la taxonomie, puis les métadonnées. La présentation du corpus est accompagné des requêtes CMQL permettant d'obtenir les informations, d'une part à titre d'exemple de requêtes, d'autre part pour montrer que les informations générales d'un corpus de cartes pourrait être obtenues de manière automatisée grâce à un ensemble de requêtes CMQL simples.

Les cartes, concepts et influences

Ce corpus contient 53 cartes cognitives.

```
SELECT COUNT(DISTINCT ?m) FROM ALL WHERE{
IsInMap(?m,?c)
}
```

COUNT(?m)
53

Chaque carte comprend entre 10 et 29 concepts.

```
SELECT ?m,COUNT(?c) FROM ALL WHERE{
IsInMap(?m,?c)
} GROUPBY ?m ORDERBY COUNT(?c) DESC
```

?m	COUNT(?c)
PB11	29
LS04	27
LC12	23
LT12	22
LT03	22
SG14	22
SN11	21
LC07	21
...	...

La colonne de gauche contient les noms des cartes cognitives, leur nom est un code indiquant le port d'attache avec les deux premières lettres (par exemple LS pour les Sables), la génération avec la troisième caractère (1 pour les pêcheurs en activité, 0 pour les pêcheurs retraités) et un indice d'identification neutre pour le dernier caractère.

Chaque carte comprend entre 10 et 35 influences.

```
SELECT ?m,COUNT(?p) FROM ALL WHERE{
Length(?m,?p,1)
} GROUPBY ?m ORDERBY COUNT(?p) DESC
```

?m	COUNT(?p)
PB11	35
LT03	32
LT12	29
LS04	28
LS01	28
SN11	27
SG14	27
LS08	27
...	...

L'ensemble de valeurs utilisé dans ce corpus est le suivant : {-4,-3,-2,-1,1,2,3,4}. La requête suivante donne une idée de l'ensemble de valeurs en montrant la répartition des valeurs d'influence.

```
SELECT ?i, COUNT(?m) FROM ALL WHERE{
DirectValue(?m,?c1,?c2,?i)
} GROUPBY ?i ORDERBY COUNT(?m) DESC
```

?i	COUNT(?m)
4	423
3	311
2	158
1	46
-3	34
-4	27
-2	15
$\varnothing$	8
-1	6

Cette requête ne donne pas forcément toutes les valeurs de l'ensemble, par exemple si une valeur n'est utilisée dans aucune carte, mais peut en donner une idée. On remarque que certaines influences n'ont pas été étiquetées, c'est pourquoi la valeur $\varnothing$ apparaît à 8 reprises. Le modèle ne permet normalement pas cela, mais cette contrainte avait été relâchée lors de la construction des cartes.

Quels sont les concepts qui sont présents dans toutes les cartes.

```
SELECT ?c FROM ALL WHERE{ ALL ?m (IsInMap(?m,?c))}
```

?c

Aucun concept est retrouvé sur toutes les cartes de l'ensemble. Les cartes semblent donc a priori très variées, ce qui donne d'autant plus d'importance à la taxonomie qui permet de faire le lien entre les concepts des différentes cartes.

La taxonomie

La taxonomie ordonne les concepts par une relation de type « est une sorte de ». Elle est utilisée pour la construction des cartes cognitives durant les entretiens, mais aussi durant l'exploitation des cartes. Dans le cadre du projet Kifanlo, l'élaboration de la taxonomie a été effectuée lors d'une colaboration entre les chercheurs en géographie et les chercheurs en informatique du laboratoire LS2N à Nantes [Har17].

La taxonomie comporte 641 concepts.

```
SELECT COUNT(DISTINCT ?c) FROM ALL WHERE{
KindOf(?c,?c)
}
```

COUNT(?c)
641

Les concepts les plus généraux donnent une idée du sens des concepts de la taxonomie.

```
SELECT ?c, COUNT(?c1) FROM ALL WHERE{
KindOf(?c1,?c)
AND NOT(KindOf(?c,?c2) AND ?c != ?c2)
} GROUPBY ?c ORDERBY COUNT(?c1) DESC
```

?c	COUNT(?c1)
FacteurEconomique	199
DescriptionMétier	180
FacteurHumain	101
Externalite	54
Tactiques	34
Temps	34
Reglement	32

La colonne de gauche comporte les 7 concepts les plus généraux de la taxonomie, tous les concepts sont donc des spécialisations de ces concepts. La colonne de droite représente le nombre de concepts que généralisent les 7 concepts les plus généraux. Par exemple 199 concepts sont des spécialisations de `FacteurEconomique`, comme `Investissement`, `Bénéfice`...

Pour avoir une idée de la profondeur de la taxonomie, on peut s'intéresser à la proportion de concepts élémentaires dans la taxonomie, c'est-à-dire des concepts les plus spécialisés. Il y en a 441, soit 69%, 31% sont donc des concepts qui généralisent d'autres concepts.

```
SELECT COUNT(DISTINCT ?c) FROM ALL WHERE{
ElemOf(?c,?c1)
}
```

COUNT(?c)
441

Pour pouvoir exploiter un ensemble de cartes cognitives taxonomiques, il est important de savoir de quel type sont les cartes. Une carte cognitive taxonomique contrainte est une carte qui ne contient que des concepts élémentaires.

```
SELECT COUNT(DISTINCT ?m) FROM ALL WHERE{
IsInMap(?m,?c) AND NOT(KindOf(?c2,?c) AND ?c2 != ?c)
}
```

count
0

Aucune carte de ce corpus ne contient que des concepts élémentaires, elles ne sont donc pas contraintes.

Étant donné la grande quantité de concepts non-élémentaires dans la taxonomie, on peut se demander si les cartes contiennent des concepts comparables (concepts dont l'un étant une spécialisation de l'autre).

```
SELECT COUNT(DISTINCT ?m) FROM ALL WHERE{
IsInMap(?m,?c1) AND IsInMap(?m,?c2)
AND KindOf(?c1,?c2) AND ?c1 != ?c2
}
```

count
0

Aucun carte ne contient deux concepts qui sont comparables. Les cartes de ce corpus sont donc des cartes cognitives non contraintes et qui contiennent pas de concepts comparables.

Pour savoir à quel point la taxonomie est exploitée dans les cartes, il est possible de regarder la proportion des concepts de la taxonomie qui sont utilisés dans les cartes.

```
SELECT COUNT(DISTINCT ?c) FROM ALL WHERE{ IsInMap(?m,?c) }
```

count
390

Un peu plus de la moitié des concepts de la taxonomie sont exploités dans les cartes (60%).

Les métadonnées

Ce corpus de cartes cognitives contient des métadonnées, qui renseignent sur le pêcheur auquel est attribué une carte cognitive. Ces métadonnées sont d'une grande importance lors de l'analyse du corpus, afin de les prendre en compte dans le langage de requête nous leur attribuons une primitive : la primitive Meta. À chaque carte est associé un ensemble de couples clé-valeur, les métadonnées sont donc représentées par des triplets carte-clé-valeur.

La primitive Meta permet d'accéder à des connaissances extérieures au modèle, c'est-à-dire aux métadonnées, et de les prendre en compte dans le langage de requête. Cette primitive lie une carte à une clé et une valeur.

Définition 50 (Primitive Meta)**.** *Soit $\mathcal{S}$ un ensemble de cartes cognitives ontologiques définies sur l'ensemble de valeurs I, la taxonomie $T = (C, \leqslant)$ et l'ontologie spatio-temporelle $\mathcal{O}_{\text{st}}$. Soit $\mathcal{M}_{\mathcal{S}}$ l'ensemble des métadonnées de $\mathcal{S}$ qui est composé de triplets (carte,clé,valeur). Soit $\mathcal{M}_{\mathcal{S}}.keys$ l'ensemble des clés de $\mathcal{M}_{\mathcal{S}}$ et $\mathcal{M}_{\mathcal{S}}.values$ l'ensemble des valeurs de $\mathcal{M}_{\mathcal{S}}$. La primitive $Meta(m : \mathcal{S}, key : \mathcal{M}_{\mathcal{S}}.keys, val : \mathcal{M}_{\mathcal{S}}.values)$ est une primitive telle que val est la valeur de la clé key pour la carte m. Sa valeur est l'ensemble $\mathcal{M}_{\mathcal{S}}$.*

Les métadonnées du corpus de Kifanlo comportent 10 clés.

```
SELECT ?col, COUNT(?map) FROM ALL WHERE{
Meta(?map,?clé,?val)
} GROUPBY ?clé
```

?clé	COUNT(?map)
carte	53
localite	53
metier 1	53
metier 2	53
metier 3	53
indice	53
navire	53
periode	53
port	53
taille	53

Dans les métadonnées du corpus de Kifanlo le même ensemble de clé décrit les différentes cartes. La première clé renseigne le nom de la carte cognitive. Les autres clés concernent le port d'attache, la période du pêcheur, le métier, le nom et la taille du navire...

La période du pêcheur fait référence à la génération du pêcheur, 30 cartes proviennent de pêcheurs en activité dans les années 1970 et 23 cartes proviennent de pêcheurs en activité en 2010.

```
SELECT COUNT(?map), ?value FROM ALL WHERE{
Meta(?map,periode,?value)
} GROUPBY ?value ORDERBY COUNT(?map)
```

?value	COUNT(?map)
1970	30
2010	23

La carte cognitive LS12 de la figure 5.1 est la carte d'un pêcheur en activité en 2010 et rattaché au port des Sables d'Olonne. Cette carte représente une partie de la stratégie de pêche de ce pêcheur, elle contient par exemple des connaissances comme le fait que le patron connaisse bien son métier influence le choix d'avoir un navire de moins de 12 mètres. Elle contient 12 concepts et 11 influences, c'est une petite carte sur laquelle s'appuient les quelques requêtes plus spécifiques qui suivent.

La requête suivante interroge la carte LS12, pour donner une idée des types de concepts présents dans la carte en les reportant sur les concepts les plus généraux.

```
SELECT ?gc, COUNT(?c) FROM LS12 WHERE{
IsInMap(?m,?c) AND
KindOf(?c, ?gc) AND
NOT(KindOf(?gc,?x) AND ?gc != ?x)
} GROUPBY ?gc ORDERBY COUNT(?c) DESC
```

?gc	COUNT(?c)
FacteurHumain	4
DescriptionMétier	3
FacteurEconomique	2
Externalite	1
Tactiques	1
Temps	1

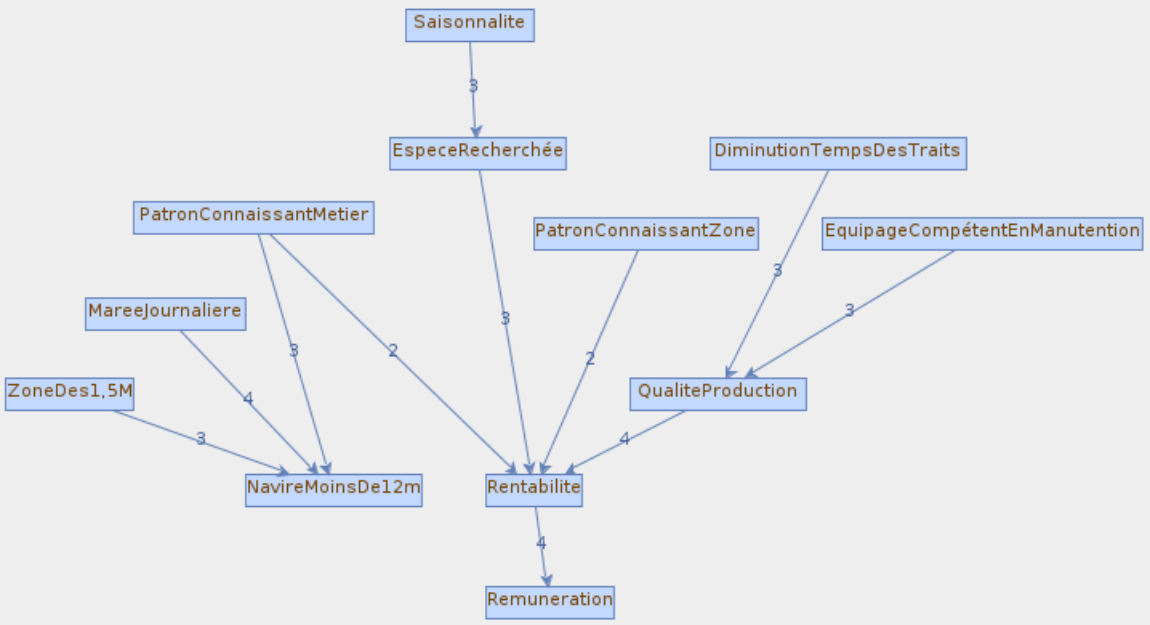


FIGURE 5.1 – Carte Cognitive LS12, issue du projet Kifanlo

Cette carte contient 4 concepts qui sont des types de `FacteurHumain`, 3 types de `DescriptionMetier`...

La requête suivante interroge quels sont les concepts qui influencent, directement ou indirectement, la rentabilité et le choix d'un navire de moins de 12 mètres.

```
SELECT ?c FROM LS12 WHERE{
Path(?m,?c,NavireMoinsDe12m,?p1) AND
Path(?m,?c,Rentabilite,?p2)
}
```

?c
PatronConnaissantMetier

Seul le concept `PatronConnaissantMétier` influence à la fois la rentabilité et le choix d'un navire de moins de 12 mètres.

La requête suivante interroge quelles sortes de tactiques influencent positivement la rentabilité.

```
SELECT ?c FROM LS12 WHERE{
KindOf(?c,Tactiques) AND
Value(?m,?c,Rentabilite, ?i) AND
?i >0
}
```

?c
DiminutionTempsDesTraits

D'après ce pêcheur, la diminution des temps de traits influence positivement la rentabilité.

5.1.2 Construction de cartes cognitives ontologiques

C'est au cours du projet kifanlo que le problème d'une bonne prise en compte des aspects temporels et spatiaux est apparu, ce qui a conduit aux propositions présentes dans cette thèse. Le modèle des cartes cognitives ontologiques comporte une ontologie spatio-temporelle et des assertions spatio-temporelles qui permettent de caractériser les concepts des cartes cognitives taxonomiques. Pour réaliser les tests de ce modèle, le corpus du projet Kifanlo a été utilisé mais a dû être modifié a posteriori étant donné qu'il s'agit d'un ensemble de cartes cognitives taxonomiques. Cela limite néanmoins les tests effectués puisqu'il aurait fallu prendre en compte les extensions du modèle dès la construction des cartes voire plus tôt, durant la préparation de la taxonomie et de l'ontologie spatio-temporelle. Les changements effectués sur le corpus sont décrits dans cette partie, avant de présenter des exemples de requêtes utilisant les primitives d'accès aux connaissances spatio-temporelles.

Modifications du corpus effectuées

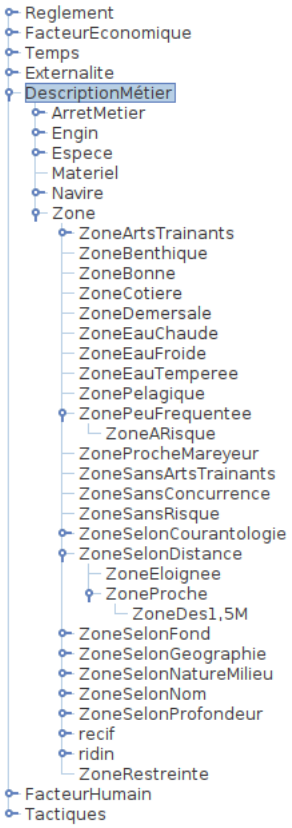
Appelons la version du corpus présenté précédemment la version 1 (V1), celle issue des changements la version 2 (V2). Les connaissances temporelles et spatiales n'ayant pas pu être récupérées directement auprès des pêcheurs il a fallu les récupérer autrement. D'une part depuis le corpus lui même, en cherchant des connaissances ayant un sens spatio-temporel bien qu'elles ne soient pas représentées en tant que telles. D'autre part, en allant chercher si besoin des connaissances en dehors du modèle, depuis toute autre source de données.

Le corpus V1 contient de nombreux concepts qui ont une sémantique temporelle ou spatiale. En prenant l'exemple de la figure 5.1, on remarque que les concepts `MareeJournaliere` et `Saisonnalite` ont une sémantique temporelle, et que le concept `ZoneDes1.5m` a une sémantique spatiale. La recherche des concepts ayant une sémantique temporelle ou spatiale s'est faite directement dans la taxonomie qui contient tous les concepts.

La taxonomie de concepts du corpus contient 34 concepts qui sont des spécialisations du concept `Temps` (figure 5.2a) et 54 qui sont des spécialisations du concept `Zone` (figure 5.2b). Le concept `ZoneDes1.5m`, apparaissant dans la carte LS12 de la figure 5.1, est d'ailleurs une spécialisation du concept `Zone`. En plus de ces concepts, d'autres concepts qui ne sont ni des spécialisations du concept `Temps` ni du concept `Zone` ont également un sémantique spatiale ou temporelle, c'est par exemple le cas du concept `TempsSortie` qui est un type de `Reglement` et des concept `tempsDeSommeilCourt` et `TempsDeSommeilLong` qui sont des types de `Tactiques`.

L'approche consiste à identifier ces concepts, puis à transférer les connaissances spatio-temporelles depuis les concepts vers une ontologie spatio-temporelle. Pour cela les concepts sont supprimés de la taxonomie et quand ils apparaissent dans les cartes cognitivesn, ils sont remplacés par un concept plus général. Une entité spatiale ou temporelle caractérisera alors le concept de la carte en le liant à l'information spatio-temporelle adéquate dans l'ontologie

(a) Concepts temporels de la taxonomie.



(b) Concepts spatiaux de la taxonomie.

FIGURE 5.2 – Taxonomie du corpus Kifanlo V1

spatio-temporelle.

Prenons le cas des quatre concepts spécialisés du concept `Maree` [1] (figure 5.2a), dont `MareeJournaliere` qui apparaît dans la carte LS12 de la figure 5.1 fait partie. Ces quatre concepts sont supprimés de la taxonomie, et des intervalles périodiques les représentant sont ajoutés dans l'ontologie spatio-temporelle avec les assertions temporelles de la figure 5.3.

Les quatre concepts supprimés, quand ils apparaissaient dans les cartes, sont remplacés par le concept plus général `Maree`. Une assertion temporelle est ajoutée à la carte pour caractériser le noeud étiqueté par `Marée` comme par exemple (T_LS12_Maree, $=_T$, MareeJournaliere).

Cela a permis de réduire la taille de la taxonomie, en transférant une partie des connaissances qu'elle contenait vers l'ontologie spatio-temporelle qui peut représenter ces connaissances de manière plus adaptée. L'ontologie spatio-temporelle a également été complétée par des informations extérieures au modèle comme par exemple pour savoir quand se trouve la saison du crabe ou de la crevette. Ces informations ont pu être facilement récupérées sur les

1. Il est rappelé que le terme « marée » signifie « Une session de pêche » dans le jargon de la pêche et non la montée et descente périodique des eaux.

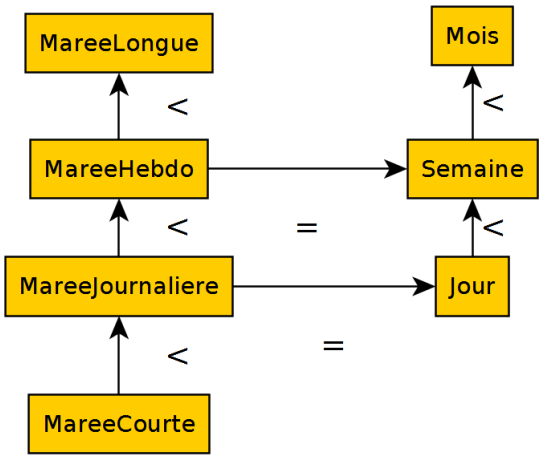


FIGURE 5.3 – Assertions temporelles ajoutées.

sites des organisations de pêcheurs comme *Les pêcheurs de Bretagne*[2].

Le corpus V2 ainsi obtenu, contient une taxonomie de 580 concepts, soit 61 de moins que le corpus V1 qui en contient 641. Le concept de `Temps` ne généralise plus que 7 concepts dans la taxonomie.

```
SELECT DISTINCT ?c FROM ALL WHERE{
KindOf(?c,Temps)
}
```

?c
Temps
CalendrierAnnuel
Maree
CalendrierAnnuelAlterne
HeureDeLaJournee
CalendierAnnuelFixe
Saison
Campagne

Bien sûr toutes les connaissances n'ont pu être modélisées, d'une part car ce sont des modifications a posteriori, ainsi le concept `HeureDeLaJournee` ne peut pas être pris en compte car il aurait fallu obtenir durant l'entretien l'information sur ce qui est influencé quand on pêche à telle heure ou à telle heure. D'autre part car certaines connaissances temporelles ne sont pas représentables par des intervalles périodiques, bien qu'ils puissent modéliser la plupart des connaissances temporelles du modèle. Par exemple les dates de mise en application des réglementations, qui sont des instants, ne sont pas représentables par des intervalles périodiques. Concernant les connaissances spatiales, la plupart n'ont pu être prises en compte, principalement par manque de connaissances expertes : comme par exemple l'emplacement des zones de plateau, des zones sableuses, des zones pélagiques ou celles à courant... Par ailleurs, les assertions spatio-temporelles ajoutées sont déduites du sens des concepts, mais peuvent

2. `https://www.pecheursdebretagne.eu/cest-la-saison/`

cependant être erronées. Il est par exemple possible que le concept `MareeCourte` fasse référence à une marée plus longue que `MareeJournaliere`, en signifiant seulement « Une marée plus courte que d'habitude ». En somme, les modifications effectuées sont très limitées pour les prendre en compte dans les analyses du corpus, il aurait fallu compléter les entretiens pour compléter les cartes, cee qui n'a pas été possible. Ces modifications sont cependant suffisantes pour tester le modèle des cartes cognitives ontologiques et les primitives d'accès aux connaissances spatio-temporelles.

Primitives spatio-temporelles

Les primitives spatio-temporelles permettent l'accès aux connaissances temporelles et spatiales du modèle. Les primitive TimeInfo et SpaceInfo permettent d'accéder aux intervalles périodiques et entités spatiales qui caractérisent les noeuds des cartes. Chaque noeud des cartes a un intervalle périodique et une entité spatiale qui lui sont associés, le corpus ayant un total de 896 noeuds dans les différentes cartes, il y a autant d'intervalles périodiques et entités spatiales. Les primitives CompareTime et CompareSpace permettent respectivement d'accéder à toutes les assertions temporelles et spatiales du modèle, y compris les assertions inférées.

Il y a 60 intervalles périodiques associés aux noeuds des cartes qui sont présents dans au moins une assertion temporelle.

```
SELECT DISTINCT ?e1 FROM ALL WHERE{
TimeInfo(?m,?c,?e1)
AND CompareTime(?e1,?p,?e2)
}
```

?e1
T_YE15_Peche
T_YE15_Maree
T_YE14_DiminutionPrixCriée
T_YE14_ConcurrencePélagique
T_YE13_Maree
T_YE12_Maree
T_YE02_RythmeDeTravail
T_YE02_Maree
T_SN12_Surpêche
...

Cela fait en moyenne un peu plus d'un concept par carte qui est caractérisé temporellement. Le concept de `Maree` revient le plus parmi les concepts caractérisés temporellement.

Il y a 44 entités spatiales associées aux noeuds des cartes qui sont présents dans au moins une assertion spatiale.

```
SELECT DISTINCT ?e1 FROM ALL WHERE{
SpaceInfo(?m,?c,?e1)
AND CompareSpace(?e1,?p,?e2)
}
```

?e1
S_YE13_Zone
S_YE12_Zone
S_YE05_Vente
S_YE04_Zone
S_SG14_Peche
S_SG13_RéglementationNavire
S_SG13_EspecePresente
S_SG06_Zone
S_SG06_Navire
...

Cela fait en moyenne un peu moins d'un concept par carte qui est caractérisé spatialement. Les concepts de `Zone` et de `Peche` reviennent le plus parmi les concepts caractérisés spatialement.

5.1.3 Exemples de requêtes CMQL

Cette dernière partie de présentation de l'exploitation d'un ensemble de cartes cognitives ontologiques concerne le langage de requête et propose un panel de requêtes diverses sur le corpus V2 du projet Kifanlo. Les requêtes suivantes mettent en avant chaque fonctionnalité du langage CMQL et sont accompagnées de leur résultat.

Quels sont les tailles des chemins qui existent de `PatronConnaissantMetier` à `Rentabilite` ?

```
SELECT ?l,COUNT(?p) FROM ALL WHERE{
Path(?m,PatronConnaissantMetier, Rentabilite,?p)
AND Length(?m,?p,?l)
} GROUPBY ?l
ORDERBY ?l DESC
```

?l	COUNT(?p)
4	2
3	11
2	15
1	5

Quels sont les types de tactiques qui influencent le plus dans les cartes ?

```
SELECT ?c,COUNT(?p)
FROM ALL
WHERE{
Path(?m,?c,?c2,?p)
AND KindOf(?c,Tactiques)
} GROUPBY ?c
ORDERBY COUNT(?p) DESC
LIMIT 15
```

?c	COUNT(?p)
ActiviteProspection	37
TraitsDeChalut	23
BolincheALaRogue	21
CommunicationEntrePatrons	19
OptimisationTemps	9
TempsDeSommeil	7
PasDeSortie	6
AdaptabiliteSaison	6
Observation_sondeur	6
EquipeMaintenanceATerre	5
ChangerDeZoneDePeche	5
AdaptabiliteEspece	5
TravailMauvaisTemps	4
NombreDeTraits	4
AdaptabiliteMeteo	3

Quels sont les couples de concepts qui s'influencent l'un l'autre ?

```
SELECT ?x,?y
FROM ALL
WHERE{
Path(?m,?x,?y,?p)
AND
Path(?m,?y,?x,?q)
} ORDERBY ?c1
LIMIT 20
```

?x	?y
Ambition	CompetitionEntreNavires
Benefices	InvestissementNouveauNavire
Benefices	InvestissementNouveauNavire
Benefices	Navire
Benefices	QualiteProduction
Benefices	ValeurProductionImportante
CompetitionEntreNavires	Ambition
InvestissementNouveauNavire	Benefices
InvestissementNouveauNavire	Navire
InvestissementNouveauNavire	ValeurProductionImportante
InvestissementNouveauNavire	Benefices
InvestissementNouveauNavire	QualiteProduction
LangoustineAuChalut	Zone
LangoustineAuChalut	RythmeDeTravail
Navire	Benefices
Navire	QualiteProduction
Navire	ValeurProductionImportante
Navire	InvestissementNouveauNavire
PasDEquipage	PasDeSortie
PasDEquipage	Peche

Quels sont les concepts qui influencent le fait de sortir ou non en mer (concept PasDeSortie), et ont-ils une influence directe ?

```
SELECT ?c1,?len
FROM ALL
WHERE{
Path(?m,?c1,PasDeSortie,?p)
AND Length(?m,?p,?len)
} ORDERBY ?len
```

?c1	?len
ArretCivelle	1
VolumeProductionImportant	1
PasDEquipage	1
BelleMeteo	1
MauvaiseMeteo	1
MatérielPourTravaillerSeul	2
LiberteDeChoix	2
PasDeGout	2
AmortissementPrêtTerminé	2
BelleMeteo	2

Quels sont les concepts influencés par les concepts types de Meteo ?

```
SELECT ?c2,COUNT(?m)
FROM ALL
WHERE{
Path(?m,?c1,?c2,?p)
AND KindOf(?c1,Meteo)
} GROUPBY ?c2
ORDERBY COUNT(?m) DESC
LIMIT 15
```

?c1	COUNT(?m)
Rentabilite	21
PrixFort	4
VolumeProductionImportant	3
PasDeSortie	3
InvestissementPossible	3
PlaisirPatron	3
Confort	2
VolumeProductionCivelle	2
AugmentationPerteMateriel	2
PasDEquipage	2
TravailMauvaisTemps	1
DebarquementCeJour	1
DiminutionProductionFlottille	1
VolumeProductionCrevettes	1
Tourisme	1

5.1.4 Bilan

Le corpus du projet Kifanlo contient 53 cartes construites par des pêcheurs et qui portent sur leur stratégie de pêche. Ce sont des cartes cognitives taxonomiques non contraintes auxquelles sont associées des métadonnées comme la période de pêche, le port d'attache, le métier ou la longueur des bateaux utilisés. La taxonomie de concepts contient un grand nombre de concepts (641) et les cartes sont assez différentes, ne présentant que peu de concepts communs. Une seconde version du corpus a été créée a posteriori pour modéliser des connaissances spatio-temporelles qui ont été récupérées de la taxonomie ainsi que de sources extérieures au corpus. L'efficacité de la modélisation spatio-temporelle nécessiterait cependant d'être évaluée dans le cadre d'un projet dans lequel les connaissances spatio-temporelles seraient intégrées dès le début du projet. Le langage de requête a lui pu être mis en oeuvre de manière efficace pour interroger les diverses connaissances du modèle. Le modèle des cartes cognitives ontologiques, comme le langage de requête CMQL et ses primitives ont été implémentés au sein du logiciel VSPCC présenté dans la partie suivante.

5.2 Système VSPCC

Cette partie vise à présenter le logiciel d'édition et d'exploitation de cartes cognitives développé dans cette thèse afin de pouvoir l'utiliser ou répliquer les résultats. Le logiciel, nommé VSPCC, a d'abord été développé par Aymeric Ledorze dans le cadre de sa thèse [LeD13] puis

d'un contrat de post-doctorat. Il a ensuite été repris et étendu dans cette thèse pour mettre en oeuvre les travaux présentés dans les chapitres précédents. Ce logiciel a également permis de pouvoir réaliser des tests sur les cartes du projet Kifanlo et vérifier que les notions introduites dans la thèse étaient applicables à des cartes issues du monde réel qui peuvent être complexes et de grande taille. Le logiciel VSPCC est disponible à l'adresse suivante : `https://sourcesup.renater.fr/projects/vspcc`.

Cette partie présente d'abord l'organisation générale de VSPCC avant les modifications apportées, puis présente l'implémentation liée aux cartes cognitives ontologiques telles qu'elles sont présentées dans le chapitre 2 et présente ensuite l'implémentation liée au langage CMQL tel qu'il est présenté dans les chapitres 3 et 4.

5.2.1 Présentation de VSPCC

VSPCC est un logiciel prototype d'édition et d'exploitation de cartes cognitives, il est décrit ici mais nous renvoyons le lecteur vers la thèse de son auteur [LeD13] pour plus de détails concernant l'organisation générale. Ce logiciel permet, par une interface graphique, d'éditer des cartes cognitives classiques, taxonomiques ou ontologiques ainsi que d'effectuer diverses opérations définies dans les thèses de Lionel Chauvin [Cha10], d'Aymeric Lerdoze [LeD13] et celle-ci.

Le logiciel VSPCC, développé en Java, est principalement composé des trois packages suivants :

— *model* : ce package contient les classes métiers, décrivant le modèle de base ainsi que divers opérations comme les influences propagées ou la vue, la synthèse et la validation de cartes.

— *io* : ce package contient les classes nécessaire à l'écriture et la lecture de cartes cognitives.

— *ui* : ce package contient les classes relatives à l'interface utilisateur.

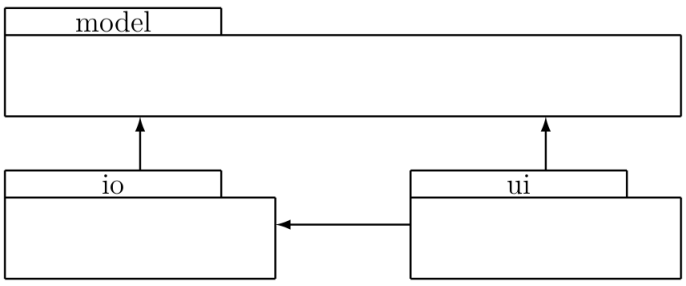


FIGURE 5.4 – Ogranisation générale des packages principaux [LeD13].

D'autres packages sont également présents comme le package *test* ou le package *controller* qui permet de gérer le modèle et les vues selon le design pattern MVC.

Package model

Le package model contient les classes de base représentant les différents objets du modèle des cartes cognitives comme les concepts, influences, chemins etc... La figure 5.5 représente le diagramme de classe du package model, en version simplifiée pour plus de lisibilité.

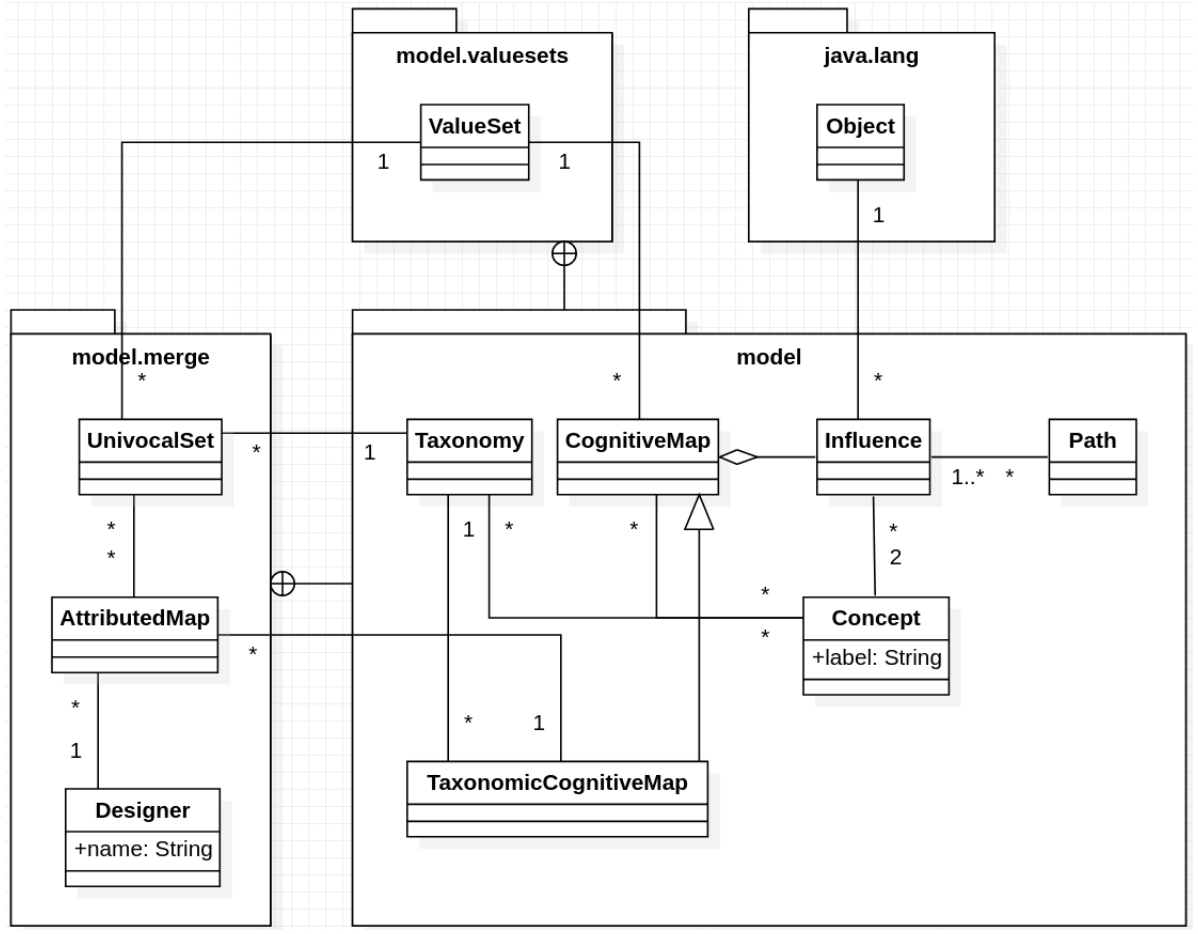


FIGURE 5.5 – Diagramme UML du package *model* [LeD13].

Le package model est composé de plusieurs sous-packages, notamment *valuesets* qui contient le code relatif à la gestion des ensembles de valeurs ou *merge* qui contient les classes nécessaires à l'opération de synthèse de cartes. D'autres sous-packages ne sont pas présents dans le digramme de la figure 5.5 comme ceux gérant la validation de cartes, les influences propagées ou les vues du design pattern MVC.

Les concepts sont représentés par des instances de la classe *Concept* et sont décrits par l'attribut *label* qui est un texte bref. Il peut cependant y avoir plusieurs fois le même concept

dans une carte, ce n'est donc pas cet attribut qui identifie le concept mais un identifiant généré automatiquement. Les influences sont représentées par des instances de la classe *Influence* qui est liée à deux instances de *Concept* qui sont les deux concepts du couple qui forme l'influence. La classe influence est liée à une instance de la classe *Objet*, qui représente la valeur de l'influence. Cette classe générique est utilisé du fait que les ensembles de valeurs peuvent être très variés (int, String, Double, enum...). Les chemins d'influence sont représentés par des instances de la classe *Path*, cette classe contient une liste d'instances d'*Influence* qui composent le chemin. Les cartes cognitives sont représentées par des instances de la classe *CognitiveMap*. Une carte est définie sur un ensemble de valeurs représenté par une instance de la classe *ValueSet*. Une carte cognitive contient plusieurs instances d'*Influences*, celles-ci sont utilisées pour calculer l'ensemble de concepts de la carte. La taxonomie est représentée par une instance de la classe *Taxonomy* qui associe à chaque concept un concept parent et un ensemble de concept enfant. La classe *Taxonomy* contient également de nombreuses méthodes donnant des informations sur la taxonomie. Une carte cognitive taxonomique est représentée par la classe *TaxonomicCognitiveMap* qui hérite de la classe *CognitiveMap*. A l'instanciation d'un objet *TaxonomicCognitiveMap*, on peut choir de contraindre la carte de différentes manières : autoriser seulement les concepts élémentaires de manière à obtenir une carte cognitive taxonomique contrainte, sinon autoriser seulement les concepts incomparables entre-eux ou non. La package *merge* qui sert à l'opération d'inférence contient la classe *UnivocalSet* qui représente un ensemble de cartes cognitives attribuées. Une carte cognitive attribuée, représentée par la classe *AttributedMap*, est composée d'une instance de *TaxonomicCognitiveMap* et d'une instance de *Designer*. La classe *Designer* représente le concepteur de la carte, elle possède un atribut *name*. En pratique nous utilisons la classe *AttributedMap* comme une carte cognitive taxonomique nommée, c'est-à-dire que l'attribut *name* de *Designer* est utilisé comme nom de la carte.

Package io

Le package io propose diverses classes de formatage des valeurs d'influences ainsi que de description d'algorithmes, mais ce package sert principalement à la gestion de la lecture et au stockage des données. Le package io propose de stocker les cartes cognitives sous différents formats comme GraphML ou BDNetworks. Le format par défaut utilisé par le logiciel pour la lecture comme pour l'écriture est un format XML [Bra+00] que nous présentons ici.

Le fichier permet de stocker un ensemble de cartes cognitives au sein d'un même fichier. Ce fichier doit contenir une ensemble de valeurs, une taxonomie et un ensemble de cartes cognitives, la racine contient donc ces trois éléments dans leur ordre respectif, qui sont les trois éléments principaux.

Le premier élément permet de renseigner l'ensemble de valeurs sur lequel est défini l'ensemble de cartes cognitives. Cet élément est composé de l'élément `<valeurs/>` qui possède

un attribut `type` indiquant le type d'ensemble qui est renseigné. Trois types sont proposés : « plusminus », « intervalle » et « enumerate ». Le type « plusminus » sert à renseigner l'ensemble de valeurs {-,+}, de la façon suivante :

```
<valeurs type="plusminus"/>
```

Le type « intervalle » sert à renseigner un ensemble de valeurs sous forme d'intervalle continu (de nombre décimaux), en utilisant ce type il faut renseigner les bornes de l'intervalle au format numérique dans les attributs `min` et `max`. L'élément *valeurs* ci-dessous renseigne l'ensemble de valeurs [-1 ;1] :

```
<valeurs type="intervalle" min="-1" max="1" />
```

Le type « enumerate » sert à renseigner un ensemble de valeurs discrètes, qu'elles soient numériques ou qualitatives. L'attribut `valueType` est obligatoire pour indiquer le type de valeurs (string, integer...). Les différentes valeurs sont à renseigner dans des éléments « <valeur/> » dans l'ordre décroissant. L'élément *valeurs* ci-dessous renseigne l'ensemble de valeurs nul < faible < moyen < fort :

```
<valeurs type="enumerate" valueType="string">
    <valeur>fort</valeur>
    <valeur>moyen</valeur>
    <valeur>faible</valeur>
    <valeur>nul</valeur>
</valeurs>
```

L'ensemble de valeur {-,+} pourrait d'ailleurs être renseigné grâce au type « enumerate » mais possède son propre type étant un ensemble classique.

Le second élément permet de stocker la taxonomie avec tous les concepts et leurs relations de généralisation. Cet élément est composée de l'élément `<taxonomie/>` qui contient une arborescence d'éléments « concept », représentant ainsi les liens de généralisation. L'élément concept contient un attribut `label` obligatoire qui permet de renseigner l'étiquette du concept, un attribut `id` qui est généré par le logiciel pour identifier le concept et des attributs `x` et `y` qui sont éventuellement renseignés par le logiciel pour indiquer la position du concept dans le composant graphique de visualisation de la taxonomie. L'élément « taxonomie » ci-dessous est un extrait de celui utilisé dans le fichier contenant les cartes du projet Kifanlo :

```
<taxonomie>
 <concept id="c0" label="FacteurHumain"  x="0.0" y="0.0">
   <concept id="c481" label="ComposanteHumaine"  x="0.0" y="0.0">
     <concept id="c554" label="Equipage"  x="0.0" y="0.0">
       <concept id="c572" label="EquipageCompetent"  x="0.0" y="0.0">
         <concept id="c579" label="EquipageCompetentEnManutention"/>
       </concept>
       <concept id="c573" label="EquipageIncompetent" x="0.0" y="0.0"/>
       <concept id="c576" label="EquipagePeuNombreux"  x="0.0" y="0.0">
```

```
10          <concept id="c578" label="PasDEquipage"  x="0.0" y="0.0"/>
11        </concept>
12        <concept id="c577" label="EquipageStable" x="0.0" y="0.0"/>
13      </concept>
14    </concept>
15  </concept>
16 </taxonomie>
```

Le troisième élément permet de stocker les cartes cognitives avec les concepts et influences utilisés. Cet élément est composée de l'élément `<cartes/>` qui contient un ensemble d'élément « carte ». Les éléments cartes possèdent trois sortes d'éléments : un élément « designer », plusieurs éléments « concept » et plusieurs éléments « influence ». L'élément « designer » permet de stocker le nom de la carte avec l'attribut obligatoire `nom` et un identifiant visuel de couleur avec l'attribut `couleur`. L'élément « concept » est renseigné comme dans l'élément taxonomie à l'exception de l'attribut label qui n'est ici plus présent. L'élement « influence » permet d'indiquer les concepts de l'influence ainsi que la valeur d'influence en utilisant les attributs `from`, `to` et `valeur`. L'élément « cartes » ci-dessous est un extrait de celui utilisé dans le fichier contenant les cartes du projet Kifanlo :

```
1  <cartes>
2    <carte>
3      <designer nom="LT11" couleur="#debe1f"/>
4      <concept id="c517" x="673.0" y="168.0"/>
5      <concept id="c227" x="109.0" y="427.0"/>
6      <influence from="c565" to="c404" valeur="3"/>
7      <influence from="c546" to="c404" valeur="3"/>
8      ...
9    </carte>
10   <carte>
11     <designer nom="LT12" couleur="#69cdf4"/>
12     <concept id="c509" x="590.0" y="590.0"/>
13     <influence from="c447" to="c100" valeur="2"/>
14     <influence from="c273" to="c175" valeur="3"/>
15     <influence from="c279" to="c100" valeur="2"/>
16     ...
17   </carte>
18   ...
19 </cartes>
```

Package ui

Le package ui contient différentes classes relatives à l'interface graphique du logiciel VSPCC, elles utilisent principalement la bibliothèque graphique Java Swing[3]. Le package ui est organisé en sous-packages qui correspondent globalement aux sous-packages du package

3. `https://docs.oracle.com/javase/6/docs/api/javax/swing/package-summary.html`

model, il contient en plus un sous-package ui.widget qui regroupe un ensemble de composants réutilisables et un sous-package ui.graph qui contient des composants relatifs à la visualisation de graphes. Les composants de ui.graph sont d'ailleurs pour la plupart des implémentations de la bibliothèque jgraph [4] utilisée pour la visualisation de graphes.

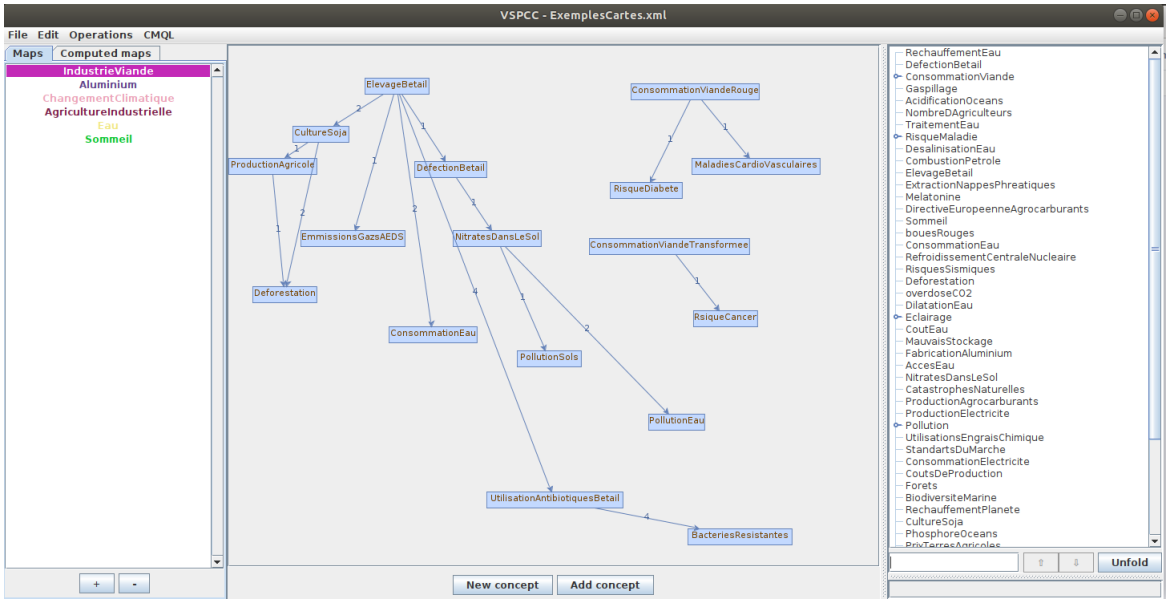


FIGURE 5.6 – Interface graphique du logiciel VSPCC.

La figure 5.6 présente la fenêtre principale du logiciel VSPCC. Sur la gauche est représenté l'ensemble des cartes, sur la droite est représentée la taxonomie de concepts et le centre contient la carte cognitive active. La taxonomie est ici présentée avec un JTree mais peut également être présentée avec un graphe sous la carte cognitive. Le menu du haut de la fenêtre propose divers onglets comme la gestion des fichiers, l'édition des cartes qui comprend des statistiques sur le modèle et les opérations du modèle telles que la synthèse ou les vues de cartes cognitives.

5.2.2 Cartes cognitives ontologiques

Cette partie décrit l'implémentation du modèle des cartes cognitives ontologiques présentées dans le chapitre 2. Il est présenté en premier lieu l'implémentation générale des différents objets tels que les assertions temporelles ou l'ontologie spatio-temporelle. L'ontologie owl qui a été intégrée est ensuite présentée avant d'expliquer le format de stockage des connaissances spatio-temporelles et l'interface utilisateur permettant de visualiser ces connaissances.

4. http://www.jgraph.com

Package model.stObjects

Le package model.stObjects contient les différentes classes nécessaires à la modélisation des cartes cognitives ontologiques. La figure 5.7 représente le diagramme de classes du package model.stObjects.

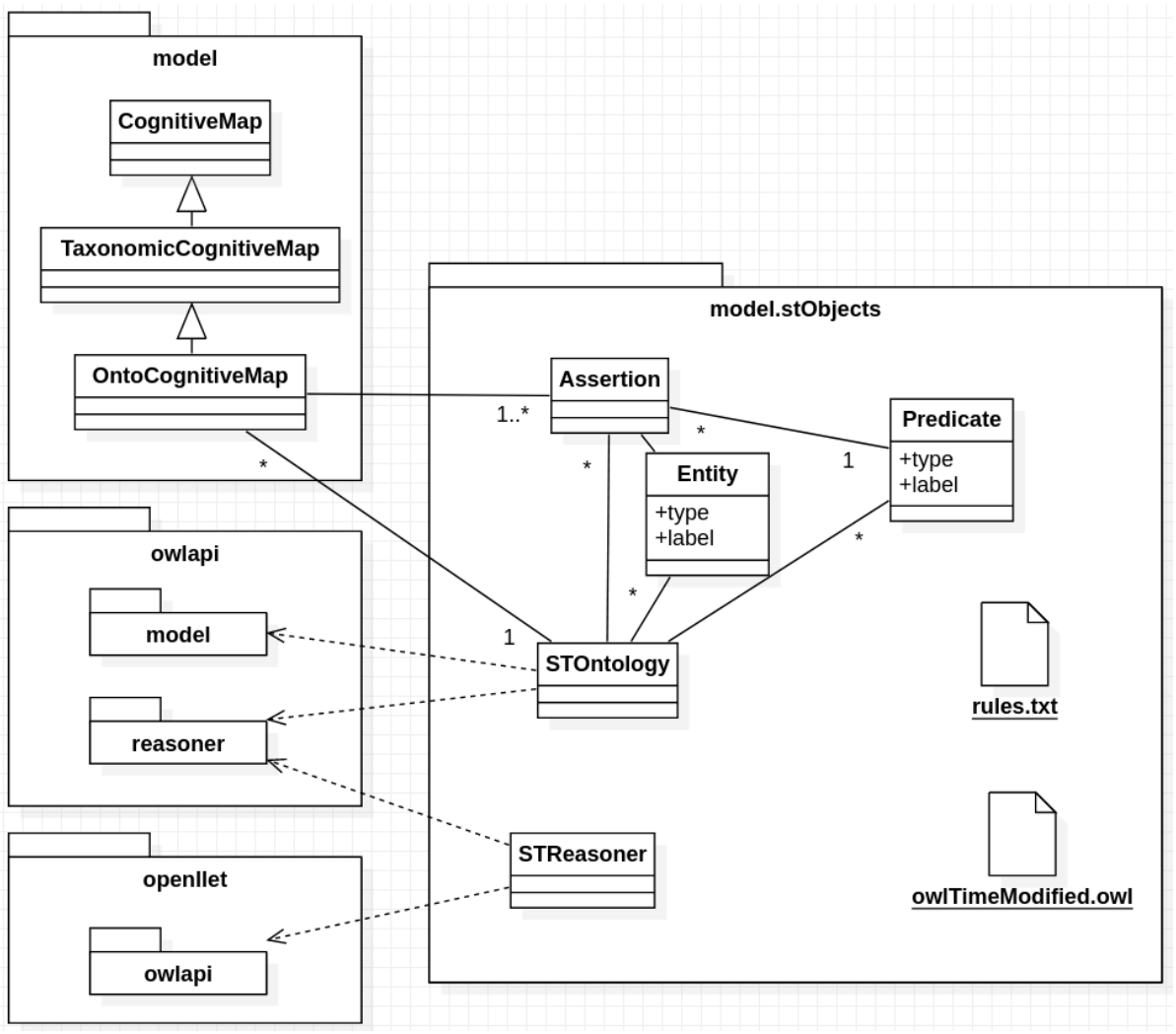


FIGURE 5.7 – Ogranisation générale du package model.stObjects.

Les connaissances spatio-temporelles sont représentées par des assertions qui sont des triplets (entité, prédicat, entité). Ces assertions sont implémentées dans la classe *Assertion* qui contient deux entités, représentées par la classe *Entity* et un prédicat représenté par la classe *Predicate*. Les classes *Entity* et *Predicate* contiennent toutes les deux un label et un type indiquant s'il s'agit d'un composant spatial ou temporel. Une carte cognitive ontologique est une carte cognitive taxonomique étendue et définie sur une une ontologie spatio-temporelle.

La classe *OntoCognitiveMap* du package *model* représente une carte cognitive ontologique, c'est une sous classe de *TaxonomicCognitiveMap* et donc de *CognitiveMap*, elle contient un ensemble d'assertions et une ontologie spatio-temporelle. L'ontologie spatio-temporelle est représentée par la classe *STOntology*, elle contient également un ensemble d'assertions. Deux bibliothèques sont utilisées pour manipuler l'ontologie : la première est owlapi [5], c'est l'outil de développement le plus important concernant OWL [Hit+09], cette bibliothèque est d'ailleurs la base du logiciel protégé [6]. owlapi permet d'instancier l'ontologie du fichier owlTimeModified.owl qui contient les différentes classes, propriétés et les assertions par défaut comme les mois ou les saisons. owlapi est aussi utilisée pour instancier les règles SWRL du fichier rules.txt. La seconde bibliothèque est openllet [7], c'est un raisonneur OWL2 basé sur Pellet [Hit+09 ; Sir+07]. openllet contient une interface pour owlapi, rendant l'utilisation de ces deux bibliothèques aisée. La classe *STReasoner* contient la gestion des inférences, elle se base sur ces deux bibliothèques.

L'ontologie utilisée ainsi que les règles d'inférence sont disponibles en annexes.

Stockage des connaissances

Les connaissances spatio-temporelles sont stockées au sein du même fichier XML que les autres connaissances. Pour cela un nouvel élément permet de renseigner des entités temporelles et spatiales de l'ontologie ainsi que des assertions temporelles ou spatiales. Cet élément est contenu par l'élément *grapheSpatioTemporel*, il contient des éléments *entite* et *assertion*. Les éléments entite contiennent un attribut *type* permettant de différencier les entités spatiales des entités temporelles et un attribut *label* correspondant au nom de l'entité. Les éléments assertion sont composés de trois attributs : un attribut *e1* et un attribut *e2* correspondant aux entités de l'assertion et un attribut *label* correspondant au prédicat.

```
<grapheSpatioTemporel>
    <entite type="SPACIALFEATURE" label="Continent" x="250.0" y="300.0"/>
    <entite type="TEMPORALENTITY" label="An" x="910.0" y="40.0"/>
    <entite type="SPACIALFEATURE" label="ZoneCotiere" x="30.0" y="290.0"/>
    <entite type="SPACIALFEATURE" label="IleDYeu" x="570.0" y="440.0"/>
    <entite type="TEMPORALENTITY" label="TraitCourt" x="20.0" y="50.0"/>
    <entite type="TEMPORALENTITY" label="SommeilLong" x="180.0" y="110.0"/>
    <assertion e1="ZoneCotiere" e2="Continent" label="closeTo"/>
    <assertion e1="TraitCourt" e2="TraitLong" label="lesserThan"/>
    <assertion e1="MareeJournaliere" e2="Jour" label="eq"/>
    <assertion e1="MareeJournaliere" e2="MareeHebdomadaire" label="lesserThan"/>
</grapheSpatioTemporel>
```

Les assertions qui sont contenues dans les cartes sont stockées directement dans l'élément « cartes » dans les éléments *concept* concernés. Un élément *entiteTemporelle* ou *spatialEntity*

5. http://owlapi.sourceforge.net/
6. https://protege.stanford.edu/
7. https://github.com/Galigator/openllet

est placé dans l'élément concept. Ces éléments peuvent alors contenir un élément *axiom* qui contiendra le prédicat et la seconde entité de l'assertion au travers des attributs *pred* et *obj*. L'exemple ci-dessous illustre le stockage des assertions spatio-temporelles des cartes :

```
<carte>
 <designer nom="LT11" couleur="#debe1f"/>
 <concept id="c556" x="792.0" y="242.0">
    <entiteTemporelle>
        <axiom pred="greaterThan" obj="An"/>
    </entiteTemporelle>
 </concept>
 <concept id="c11" x="450.0" y="379.0">
    <entiteTemporelle>
        <axiom pred="eq" obj="SaisonSeiche"/>
        <axiom pred="during" obj="Ete"/>
    </entiteTemporelle>
 </concept>
</carte>
```

Interface utilisateur

La figure 5.8 présente l'interface graphique du logiciel VSPCC qui prend en compte les connaissances du modèle des cartes cognitives ontologiques. Ainsi un panel (en bas à droite) a été ajouté pour représenter les assertions spatio-temporelles de l'ontologie sur laquelle est définie la carte courante. Les entités spatiales sont en bleu tandis que les entités temporelles, c'est-à-dire les intervalles périodiques, sont en jaune.

Les assertions contenues dans les cartes peuvent être visualisées sous les concepts comme présenté au chapitre 3, ou à la sélection des concepts.

5.2.3 Le langage de requête CMQL

Cette partie décrit l'implémentation du langage de requête CMQL présenté dans les chapitres 3 et 4. Le code concernant le langage de requête a été placé à part dans un package spécifique : le package cmql. Cette partie présente d'abord l'organisation générale du package cmql avant de présenter le fonctionnement global de l'exécution d'une requête puis de présenter l'interface utilisée.

Package CMQL

Le package cmql contient de nombreuses classes qui permettent de faire fonctionner le langage de requête CMQL. Nous présentons ici les classes principales de ce package en s'appuyant sur le diagramme UML de la figure 5.9.

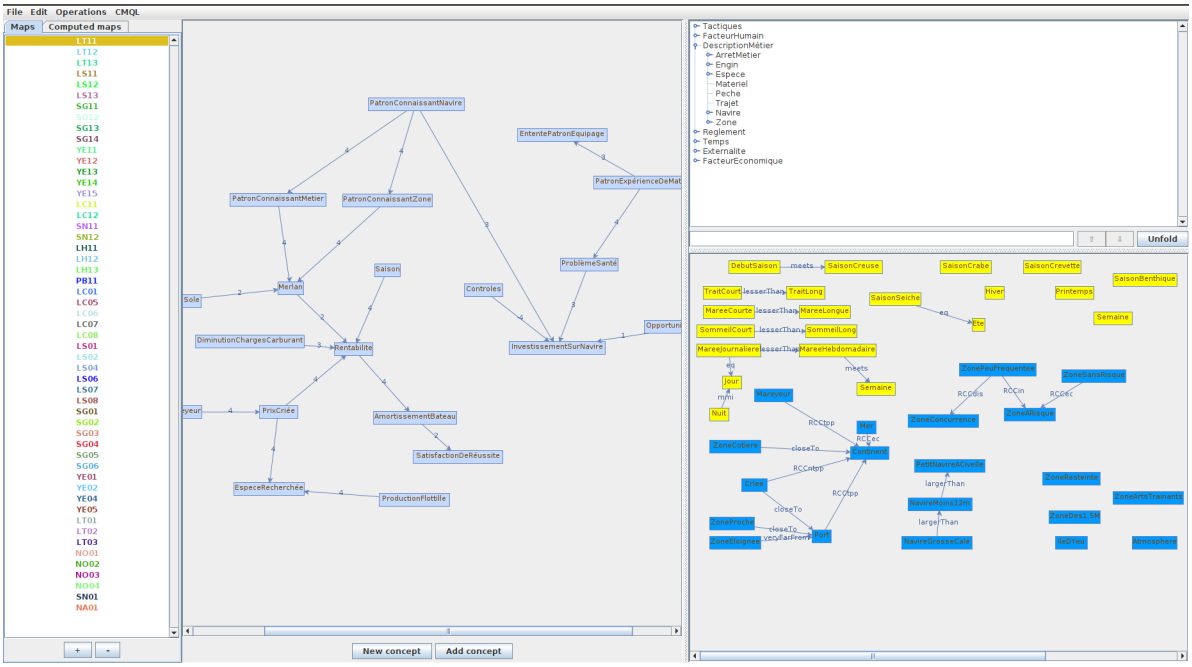


FIGURE 5.8 – Interface graphique de cartes cognitives ontologiques.

Les relations sont centrales dans le langage de requête CMQL, étant donné qu'elles forment le sens des formules et des requêtes. Une relation est ici représentée par une instance de la classe *Tuples*. Cette classe contient trois attributs principaux : *types* qui est une liste des attributs de la relation, *tuples* qui est une liste de tuples et *bValue* un booléen qui indique si la relation contient une expression à évaluer. Les tuples sont représentés individuellement comme des instances de la classe *Tuple*, qui contient les éléments des tuples. Les types, qui désignent les variables, ou plus généralement des attributs de la relation, sont représentés par des instances de la classe *VarInfo*. Cette classe contient un attribut permettant d'identifier ce qui représente l'attribut de la relation dans le langage de requête, si c'est une constante, ou une variable libre, liée. Elle contient un attribut permettant d'identifier le domaine associé à l'attribut de la relation pour typer les variables du langage CMQL. La classe *VarInfo* contient aussi l'attribut *name* permettant de nommer les variables ou attributs de la relation. La sémantique du langage CMQL étant basée sur les relations, la majorité de l'exécution des requêtes porte sur les quelques classes présentées ici. Les autres classes du diagramme permettent le fonctionnement de l'analyse et de l'exécution des requêtes.

Fonctionnement global

Le package cmql se base sur la bibliothèque ANTLR. Cette bibliothèque permet de spécifier un langage au travers de règles de syntaxe et de générer du code facilitant le traitement de ce langage. La spécification de la syntaxe est composée de deux parties, des règles de *Parser* qui

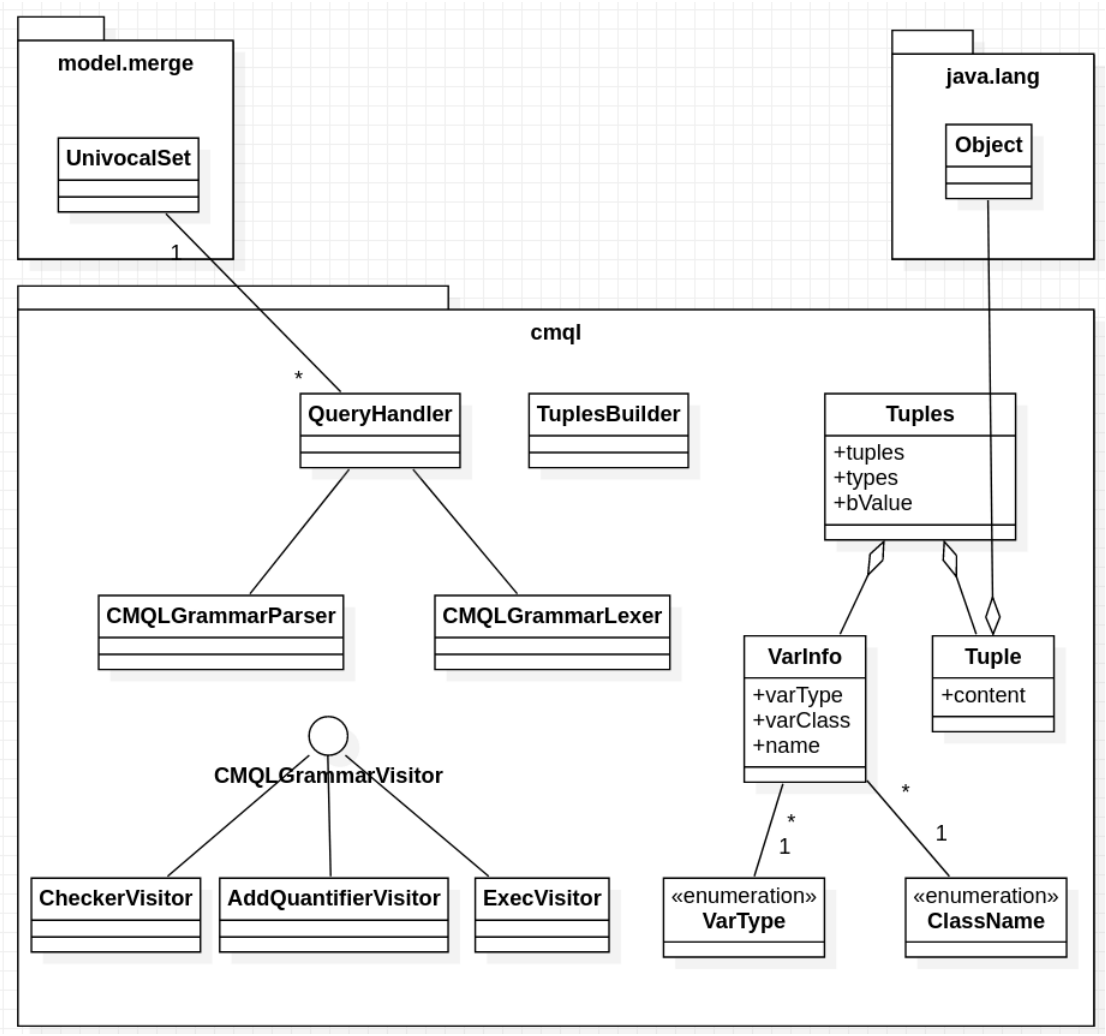


FIGURE 5.9 – Ogranisation générale du package cmql.

sont très similaires à la syntaxe présentée dans le chapitre 4, et des règles de *Lexer* ou de *Tokenisation* qui décrivent en détail les règles atomiques. Cette bibliothèque génère entre autres les classes *CMQLGrammarLexer*, *CMQLGrammarParser* qui gérent l'analyse syntaxique et l'interface *CMQLGrammarVisitor* qui facilite le parcours des arbres syntaxiques créés. Dans le package cmql, la classe *QueryHandler* gère l'intégralité du traitement des requêtes. Son fonctionnement est décrit par le diagramme de la figure 5.10.

Le gestionnaire de requête (classe *QueryHandler*) rècupère une requête sous forme de texte et retourne son résultat sous formes de tuples. Dans un premier temps une étape d'analyse textuelle est effectuée, elle est principalement gérée par la bibliothèque ANTLR. Le texte est transformé en une suite de tokens par la classe *CMQLGrammarLexer* qui va identifier les différents éléments du langage, puis les tokens sont assemblés par la classe *CMQLGrammarParser* qui va renvoyer un arbre syntaxique de la requête. Cet arbre est une instance de la

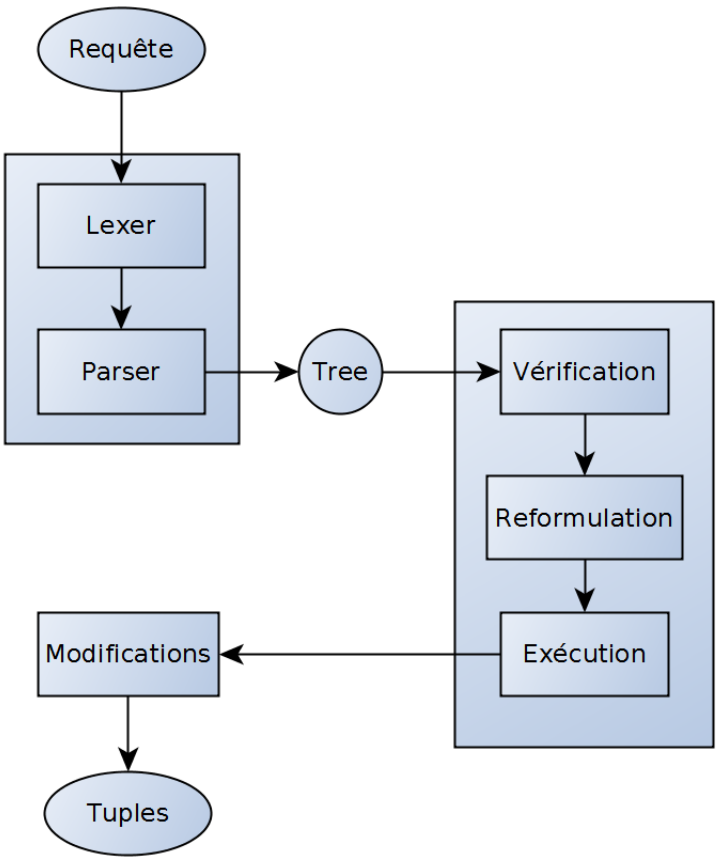


FIGURE 5.10 – Fonctionnement du traitement des requêtes CMQL.

classe org.antlr.v4.runtime.tree.ParseTree de la bibliothèque ANTLR.

Dans un second temps, une étape de traitement de l'arbre syntaxique est effectuée. Cette étape est principalement composée des trois classes *CheckerVisitor*, *AddQuantifierVisitor* et *ExecVisitor* qui sont toutes les trois des implémentations de l'interface *CMQLGrammarVisitor* générée par ANTLR. Cette interface propose un pattern visiteur pour la classe représentant l'arbre syntaxique *ParseTree*. Le CheckerVisitor s'occupe principalement de vérifier que les règles de grammaires sont bien respectées, certaines ne sont en effet pas prises en compte dans la syntaxe comme le fait que les variables libres de la requête doivent apparaître dans la formule du WHERE. Cette classe permet ainsi d'analyser quelles sont les variables et constantes utilisées, de leur associer un domaine et de vérifier que les variables sont bien quantifiées comme expliqué dans le chapitre 4. Si besoin, la classe *CheckerVisitor* fait appel au visiteur de reformulation de la classe *AddQuantifierVisitor* qui permet de modifier la requête en ajoutant les quantificateurs aux bons endroits pour les variables non quantifiées. Ces deux visiteurs, comme chacun de leurs noeuds, renvoient des chaînes de caractères qui représentent les erreurs éventuelles pour le premier et la requête reformulée pour le second. Le troisième visiteur, *ExecVisitor*, arrive en dernier et s'occupe de l'exécution de la requête. Son type de

retour est la classe *Tuples* qui contient l'ensemble des tuples. Cette classe visite les formules de la requête pour calculer leur sens. Pour cela, elle fait appel à la classe *TupleBuilder* qui permet de créer les ensembles de tuples en fonction des formules, des constantes et du sens des formules plus profondes dans l'arbre syntaxique.

Dans un dernier temps, il est possible que des modifications « post-requête » soient à effectuer comme compter, ordonner ou limiter les tuples. Ces modifications éventuelles sont détectées par le premier visiteur de traitement (*CheckerVisitor*) et sont réalisées par la classe *Modifiers* qui renvoie alors le résultat final, instance de *Tuples*.

Interface utilisateur

Une interface utilisateur Java.Swing simple a été développée pour l'utilisation de CMQL et son intégration dans VSPCC, elle se trouve principalement dans le sous-package *ui.query*. Une fenêtre est proposée pour la rédaction de requêtes et la visualisation des résultats.

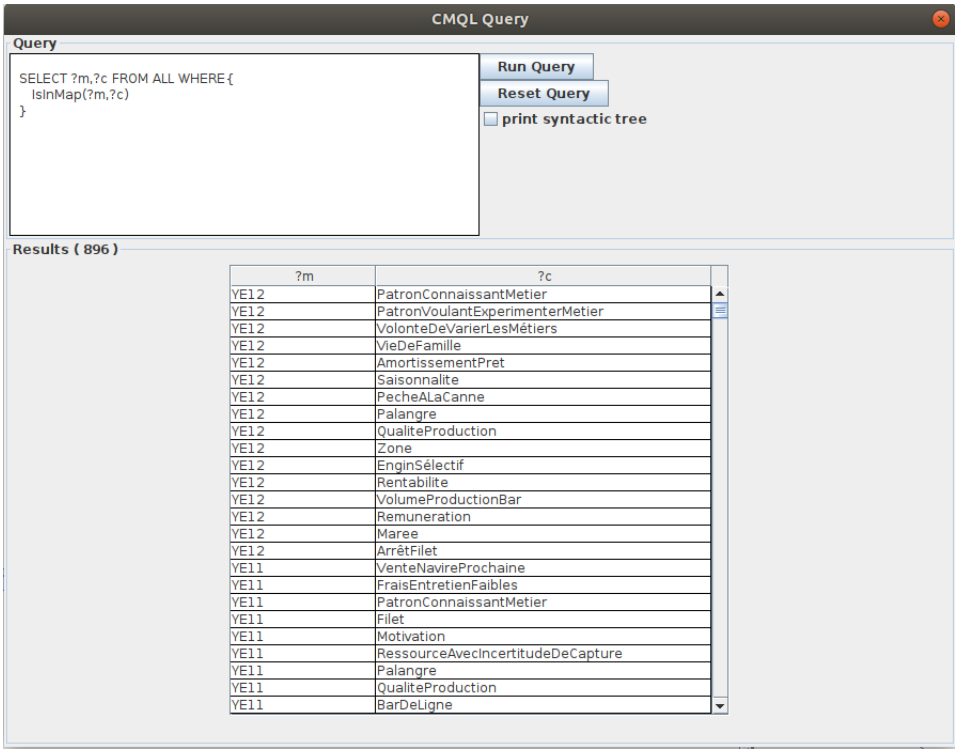


FIGURE 5.11 – Interface graphique de CMQL.

La figure 5.11 représente la fenêtre « CMQL Query », elle contient une partie *Requête* et une partie *Résultat*. La partie requête est composée d'un JTextArea pour l'écriture de la requête, d'une option permettant d'afficher l'arbre syntaxique obtenu pendant le traitement de la requête et d'un bouton pour lancer la requête. La partie résultats contient le nombre de

résultats et une JTable permettant d'afficher les tuples résultant de la requête.

5.2.4 Bilan

Le logiciel VSPCC est un logiciel d'édition et d'exploitation de cartes cognitives qui a été implémenté dans le cadre de recherches antérieures à cette thèse. Il a servi de base pour implémenter les notions introduites dans cette thèse, que ce soit le modèle des cartes cognitives ontologiques ou le langage de requêtes CMQL. Il a ensuite pu servir a valider ces travaux en les appliquant au corpus de cartes cognitives taxonomiques du projet de recherche Kifanlo.

CONCLUSION GÉNÉRALE

Les résultats de notre travail de thèse permettent de faciliter la modélisation et l'exploitation de cartes cognitives.

D'abord, le modèle de cartes cognitives ontologiques étend le modèle des cartes cognitives taxonomiques en modélisant des connaissances spatiales et temporelles. Ces connaissances sont représentées dans une ontologie spatio-temporelle qui s'appuie sur des entités spatiales, des intervalles périodiques et des prédicats de comparaison qui permettent de les comparer. Les concepts des cartes cognitives peuvent alors être caractérisés spatialement et temporellement afin de fournir une meilleure précision aux cartes cognitives.

Ensuite, notre travail de thèse propose des primitives d'accès aux connaissances pour faire face aux complexités d'analyse des cartes. Les primitives permettent de modéliser le modèle lui-même en mettant en relation les différents objets du modèle. Cette modélisation a pour but de faciliter l'accès aux connaissances du modèle. Grâce à leur syntaxe simple, elles forment une base pour que l'utilisateur puisse exprimer des requêtes sur le modèle, elles sont également facilement interprétables par la machine. Elles permettent l'accès aux connaissances du modèle classique des cartes cognitives ainsi qu'aux connaissances taxonomiques et ontologiques. Il est d'ailleurs possible de les adapter à de futures extensions du modèle.

Enfin, notre travail de thèse propose également un langage de requête, nommé CMQL, qui est basé sur les primitives d'accès et permet d'interroger le modèle des cartes cognitives ontologiques. Grâce à sa syntaxe proche de SQL, il permet d'exprimer de manière déclarative et expressive n'importe quelle connaissance à interroger. CMQL permet de se baser sur les primitives en utilisant des constantes et des variables, puis combine les primitives entre elles avec les opérateurs de la logique du premier ordre pour décrire les connaissances désirées. Un utilisateur peut alors avec aisance, interroger un ensemble de cartes cognitives ontologiques.

Les différentes contributions proposées dans cette thèse ont été implémentées au sein du logiciel VSPCC et il a été possible de les tester lors de l'exploitation d'un ensemble de cartes cognitives taxonomiques provenant d'un cas concret d'utilisation.

Pour finir, une perspective scientifique et une perspective applicative peuvent être évoquées.

En ce qui concerne la perspective scientifique, il existe quelques méthodes et métriques de comparaison de cartes cognitives dans la littérature [DSS12 ; Ede04 ; YJ16]. Ces méthodes sont toutefois assez rudimentaires, elles sont principalement centrées sur la réduction de compléxité des cartes à analyser et des métriques simples comme le nombre de sommets d'une carte. CMQL permet déjà de comparer des cartes entre elles, il serait intéressant de fournir des

métriques plus évoluées et de pouvoir les utiliser au travers de nouvelles primitives à inventer dans le langage.

En ce qui concerne la perspective applicative, les besoins issus du projet Kifanlo ont guidé le travail de cette thèse. La diffusion de VSPCC auprès dautres laboratoires de géographie nous apparaît répondre à une attente que nous souhaitons satisfaire.

LISTES DES PUBLICATIONS LIÉES À LA THÈSE

Journal à comité de lecture

— (En finalisation de relecture) Adrian ROBERT , David GENEST et Stéphane LOISEAU, « Cognitive Map Query Language for Temporal Domains », dans Agents & Artificial Intelligence.

Conférences internationales à comité de lecture

— Adrian ROBERT, David GENEST et Stéphane LOISEAU, « A Query Language for Cognitive Maps », in : *International Conference on Artificial Intelligence : Methodology, Systems, and Applications*, Springer, 2018, p. 218-227

— Adrian ROBERT, David GENEST et Stéphane LOISEAU, « The Taxonomic Cognitive Map Query Language : A General Approach to Analyse Cognitive Maps », in : *2018 IEEE 30th International Conference on Tools with Artificial Intelligence (ICTAI)*, IEEE, 2018, p. 999-1006

— Adrian ROBERT, David GENEST et Stéphane LOISEAU, « Temporal Cognitive Maps. », in : *ICAART (2)*, 2020, p. 58-68

Conférences nationales à comité de lecture

— Adrian ROBERT, David GENEST, Stéphane LOISEAU, Thomas RAIMBAULT et Brice TROUILLET, « Les cartes cognitives temporelles : modélisation et interrogation. », in : *EGC*, 2019, p. 369-370

— Adrian ROBERT, David GENEST et Stéphane LOISEAU, « Cartes Cognitives Temporelles. », in : *EGC*, 2020, p. 309-316

ANNEXES

Règles SWRL d'inférence

Les règles d'inférence présentées dans le chapitre 3 sur les primitives temporelles et spatiales sont données ici de manière exhaustive. Un premier tableau présente les règles concernant la partie temporelle et un second présente les règles concernant la partie spatiale.

Le tableau suivant donne les 179 règles d'inférence sur les assertions temporelles selon la syntaxe HRS [Hor+04] suivies de leur type OWL 2 :

n^o	rule	type
1	eq(?x1, ?x2) $\wedge$ eq(?x2, ?x3) $\Longrightarrow$ eq(?x1, ?x3)	TransitiveProperty
2	eq(?x1, ?x2) $\wedge$ m(?x2, ?x3) $\Longrightarrow$ m(?x1, ?x3)	SubPropertyChain
3	eq(?x1, ?x2) $\wedge$ mi(?x2, ?x3) $\Longrightarrow$ mi(?x1, ?x3)	SubPropertyChain
4	eq(?x1, ?x2) $\wedge$ s(?x2, ?x3) $\Longrightarrow$ s(?x1, ?x3)	SubPropertyChain
5	eq(?x1, ?x2) $\wedge$ si(?x2, ?x3) $\Longrightarrow$ si(?x1, ?x3)	SubPropertyChain
6	eq(?x1, ?x2) $\wedge$ d(?x2, ?x3) $\Longrightarrow$ d(?x1, ?x3)	SubPropertyChain
7	eq(?x1, ?x2) $\wedge$ di(?x2, ?x3) $\Longrightarrow$ di(?x1, ?x3)	SubPropertyChain
8	eq(?x1, ?x2) $\wedge$ f(?x2, ?x3) $\Longrightarrow$ f(?x1, ?x3)	SubPropertyChain
9	eq(?x1, ?x2) $\wedge$ fi(?x2, ?x3) $\Longrightarrow$ fi(?x1, ?x3)	SubPropertyChain
10	eq(?x1, ?x2) $\wedge$ o(?x2, ?x3) $\Longrightarrow$ o(?x1, ?x3)	SubPropertyChain
11	eq(?x1, ?x2) $\wedge$ oi(?x2, ?x3) $\Longrightarrow$ oi(?x1, ?x3)	SubPropertyChain
12	eq(?x1, ?x2) $\wedge$ mmi(?x2, ?x3) $\Longrightarrow$ mmi(?x1, ?x3)	SubPropertyChain
13	eq(?x1, ?x2) $\wedge$ ooi(?x2, ?x3) $\Longrightarrow$ ooi(?x1, ?x3)	SubPropertyChain
14	eq(?x1, ?x2) $\wedge$ ppi(?x2, ?x3) $\Longrightarrow$ ppi(?x1, ?x3)	SubPropertyChain
15	eq(?x1, ?x2) $\wedge$ omi(?x2, ?x3) $\Longrightarrow$ omi(?x1, ?x3)	SubPropertyChain
16	eq(?x1, ?x2) $\wedge$ moi(?x2, ?x3) $\Longrightarrow$ moi(?x1, ?x3)	SubPropertyChain
17	eq(?x1, ?x2) $\wedge$ in(?x2, ?x3) $\Longrightarrow$ in(?x1, ?x3)	SubPropertyChain
18	eq(?x1, ?x2) $\wedge$ dis(?x2, ?x3) $\Longrightarrow$ dis(?x1, ?x3)	SubPropertyChain
19	eq(?x1, ?x2) $\wedge$ lT(?x2, ?x3) $\Longrightarrow$ lT(?x1, ?x3)	SubPropertyChain
20	eq(?x1, ?x2) $\wedge$ gT(?x2, ?x3) $\Longrightarrow$ gT(?x1, ?x3)	SubPropertyChain
21	eq(?x1, ?x2) $\wedge$ isEq(?x2, ?x3) $\Longrightarrow$ isEq(?x1, ?x3)	SubPropertyChain
22	eq(?x1, ?x2) $\Longrightarrow$ eq(?x2, ?x1)	SymmetricProperty
23	m(?x1, ?x2) $\wedge$ eq(?x2, ?x3) $\Longrightarrow$ m(?x1, ?x3)	SubPropertyChain
24	m(?x1, ?x2) $\wedge$ fi(?x2, ?x3) $\Longrightarrow$ ppi(?x1, ?x3)	SubPropertyChain
25	m(?x1, ?x2) $\wedge$ di(?x2, ?x3) $\Longrightarrow$ ppi(?x1, ?x3)	SubPropertyChain

26	$m(?x1, ?x2) \wedge si(?x2, ?x3) \implies m(?x1, ?x3)$	SubPropertyChain
27	$m(?x1, ?x2) \wedge moi(?x2, ?x3) \implies d(?x1, ?x3)$	SubPropertyChain
28	$m(?x1, ?x2) \wedge ooi(?x2, ?x3) \implies d(?x1, ?x3)$	SubPropertyChain
29	$m(?x1, ?x2) \wedge mmi(?x2, ?x3) \implies f(?x1, ?x3)$	SubPropertyChain
30	$m(?x1, ?x2) \wedge omi(?x2, ?x3) \implies f(?x1, ?x3)$	SubPropertyChain
31	$m(?x1, ?x2) \implies mi(?x2, ?x1)$	InverseProperty
32	$mi(?x1, ?x2) \wedge eq(?x2, ?x3) \implies mi(?x1, ?x3)$	SubPropertyChain
33	$mi(?x1, ?x2) \wedge fi(?x2, ?x3) \implies mi(?x1, ?x3)$	SubPropertyChain
34	$mi(?x1, ?x2) \wedge di(?x2, ?x3) \implies ppi(?x1, ?x3)$	SubPropertyChain
35	$mi(?x1, ?x2) \wedge si(?x2, ?x3) \implies ppi(?x1, ?x3)$	SubPropertyChain
36	$mi(?x1, ?x2) \wedge moi(?x2, ?x3) \implies s(?x1, ?x3)$	SubPropertyChain
37	$mi(?x1, ?x2) \wedge ooi(?x2, ?x3) \implies d(?x1, ?x3)$	SubPropertyChain
38	$mi(?x1, ?x2) \wedge mmi(?x2, ?x3) \implies s(?x1, ?x3)$	SubPropertyChain
39	$mi(?x1, ?x2) \wedge omi(?x2, ?x3) \implies d(?x1, ?x3)$	SubPropertyChain
40	$mi(?x1, ?x2) \implies m(?x2, ?x1)$	InverseProperty
41	$s(?x1, ?x2) \wedge eq(?x2, ?x3) \implies s(?x1, ?x3)$	SubPropertyChain
42	$s(?x1, ?x2) \wedge s(?x2, ?x3) \implies s(?x1, ?x3)$	TransitiveProperty
43	$s(?x1, ?x2) \wedge d(?x2, ?x3) \implies d(?x1, ?x3)$	SubPropertyChain
44	$s(?x1, ?x2) \wedge f(?x2, ?x3) \implies d(?x1, ?x3)$	SubPropertyChain
45	$s(?x1, ?x2) \wedge mi(?x2, ?x3) \implies mi(?x1, ?x3)$	SubPropertyChain
46	$s(?x1, ?x2) \wedge mmi(?x2, ?x3) \implies mi(?x1, ?x3)$	SubPropertyChain
47	$s(?x1, ?x2) \wedge ppi(?x2, ?x3) \implies ppi(?x1, ?x3)$	SubPropertyChain
48	$s(?x1, ?x2) \wedge m(?x2, ?x3) \implies ppi(?x1, ?x3)$	SubPropertyChain
49	$s(?x1, ?x2) \implies si(?x2, ?x1)$	InverseProperty
50	$si(?x1, ?x2) \wedge eq(?x2, ?x3) \implies si(?x1, ?x3)$	SubPropertyChain
51	$si(?x1, ?x2) \wedge fi(?x2, ?x3) \implies di(?x1, ?x3)$	SubPropertyChain
52	$si(?x1, ?x2) \wedge di(?x2, ?x3) \implies di(?x1, ?x3)$	SubPropertyChain
53	$si(?x1, ?x2) \wedge si(?x2, ?x3) \implies si(?x1, ?x3)$	TransitiveProperty
54	$si(?x1, ?x2) \wedge moi(?x2, ?x3) \implies ooi(?x1, ?x3)$	SubPropertyChain
55	$si(?x1, ?x2) \wedge ooi(?x2, ?x3) \implies ooi(?x1, ?x3)$	SubPropertyChain
56	$si(?x1, ?x2) \wedge mmi(?x2, ?x3) \implies moi(?x1, ?x3)$	SubPropertyChain
57	$si(?x1, ?x2) \wedge omi(?x2, ?x3) \implies moi(?x1, ?x3)$	SubPropertyChain
58	$si(?x1, ?x2) \implies s(?x2, ?x1)$	InverseProperty
59	$d(?x1, ?x2) \wedge eq(?x2, ?x3) \implies d(?x1, ?x3)$	SubPropertyChain
60	$d(?x1, ?x2) \wedge s(?x2, ?x3) \implies d(?x1, ?x3)$	SubPropertyChain
61	$d(?x1, ?x2) \wedge d(?x2, ?x3) \implies d(?x1, ?x3)$	TransitiveProperty
62	$d(?x1, ?x2) \wedge f(?x2, ?x3) \implies d(?x1, ?x3)$	SubPropertyChain
63	$d(?x1, ?x2) \wedge mi(?x2, ?x3) \implies ppi(?x1, ?x3)$	SubPropertyChain
64	$d(?x1, ?x2) \wedge mmi(?x2, ?x3) \implies ppi(?x1, ?x3)$	SubPropertyChain

65	$d(?x1, ?x2) \wedge ppi(?x2, ?x3) \implies ppi(?x1, ?x3)$	SubPropertyChain
66	$d(?x1, ?x2) \wedge m(?x2, ?x3) \implies ppi(?x1, ?x3)$	SubPropertyChain
67	$d(?x1, ?x2) \implies di(?x2, ?x1)$	InverseProperty
68	$di(?x1, ?x2) \wedge eq(?x2, ?x3) \implies di(?x1, ?x3)$	SubPropertyChain
69	$di(?x1, ?x2) \wedge fi(?x2, ?x3) \implies di(?x1, ?x3)$	SubPropertyChain
70	$di(?x1, ?x2) \wedge di(?x2, ?x3) \implies di(?x1, ?x3)$	TransitiveProperty
71	$di(?x1, ?x2) \wedge si(?x2, ?x3) \implies di(?x1, ?x3)$	SubPropertyChain
72	$di(?x1, ?x2) \wedge moi(?x2, ?x3) \implies ooi(?x1, ?x3)$	SubPropertyChain
73	$di(?x1, ?x2) \wedge ooi(?x2, ?x3) \implies ooi(?x1, ?x3)$	SubPropertyChain
74	$di(?x1, ?x2) \wedge mmi(?x2, ?x3) \implies ooi(?x1, ?x3)$	SubPropertyChain
75	$di(?x1, ?x2) \wedge omi(?x2, ?x3) \implies ooi(?x1, ?x3)$	SubPropertyChain
76	$di(?x1, ?x2) \implies d(?x2, ?x1)$	InverseProperty
77	$f(?x1, ?x2) \wedge eq(?x2, ?x3) \implies f(?x1, ?x3)$	SubPropertyChain
78	$f(?x1, ?x2) \wedge s(?x2, ?x3) \implies d(?x1, ?x3)$	SubPropertyChain
79	$f(?x1, ?x2) \wedge d(?x2, ?x3) \implies d(?x1, ?x3)$	SubPropertyChain
80	$f(?x1, ?x2) \wedge f(?x2, ?x3) \implies f(?x1, ?x3)$	TransitiveProperty
81	$f(?x1, ?x2) \wedge mi(?x2, ?x3) \implies ppi(?x1, ?x3)$	SubPropertyChain
82	$f(?x1, ?x2) \wedge mmi(?x2, ?x3) \implies m(?x1, ?x3)$	SubPropertyChain
83	$f(?x1, ?x2) \wedge ppi(?x2, ?x3) \implies ppi(?x1, ?x3)$	SubPropertyChain
84	$f(?x1, ?x2) \wedge m(?x2, ?x3) \implies m(?x1, ?x3)$	SubPropertyChain
85	$f(?x1, ?x2) \implies fi(?x2, ?x1)$	InverseProperty
86	$fi(?x1, ?x2) \wedge eq(?x2, ?x3) \implies fi(?x1, ?x3)$	SubPropertyChain
87	$fi(?x1, ?x2) \wedge fi(?x2, ?x3) \implies fi(?x1, ?x3)$	TransitiveProperty
88	$fi(?x1, ?x2) \wedge di(?x2, ?x3) \implies di(?x1, ?x3)$	SubPropertyChain
89	$fi(?x1, ?x2) \wedge si(?x2, ?x3) \implies di(?x1, ?x3)$	SubPropertyChain
90	$fi(?x1, ?x2) \wedge moi(?x2, ?x3) \implies moi(?x1, ?x3)$	SubPropertyChain
91	$fi(?x1, ?x2) \wedge ooi(?x2, ?x3) \implies ooi(?x1, ?x3)$	SubPropertyChain
92	$fi(?x1, ?x2) \wedge mmi(?x2, ?x3) \implies moi(?x1, ?x3)$	SubPropertyChain
93	$fi(?x1, ?x2) \wedge omi(?x2, ?x3) \implies ooi(?x1, ?x3)$	SubPropertyChain
94	$fi(?x1, ?x2) \implies f(?x2, ?x1)$	InverseProperty
95	$o(?x1, ?x2) \wedge eq(?x2, ?x3) \implies o(?x1, ?x3)$	SubPropertyChain
96	$o(?x1, ?x2) \wedge mmi(?x2, ?x3) \implies oi(?x1, ?x3)$	SubPropertyChain
97	$o(?x1, ?x2) \implies oi(?x2, ?x1)$	InverseProperty
98	$oi(?x1, ?x2) \wedge eq(?x2, ?x3) \implies oi(?x1, ?x3)$	SubPropertyChain
99	$oi(?x1, ?x2) \wedge mmi(?x2, ?x3) \implies o(?x1, ?x3)$	SubPropertyChain
100	$oi(?x1, ?x2) \implies o(?x2, ?x1)$	InverseProperty
101	$mmi(?x1, ?x2) \wedge eq(?x2, ?x3) \implies mmi(?x1, ?x3)$	SubPropertyChain
102	$mmi(?x1, ?x2) \wedge o(?x2, ?x3) \implies oi(?x1, ?x3)$	SubPropertyChain
103	$mmi(?x1, ?x2) \wedge s(?x2, ?x3) \implies moi(?x1, ?x3)$	SubPropertyChain

104	$mmi(?x1, ?x2) \wedge d(?x2, ?x3) \implies ooi(?x1, ?x3)$	SubPropertyChain
105	$mmi(?x1, ?x2) \wedge fi(?x2, ?x3) \implies mi(?x1, ?x3)$	SubPropertyChain
106	$mmi(?x1, ?x2) \wedge f(?x2, ?x3) \implies omi(?x1, ?x3)$	SubPropertyChain
107	$mmi(?x1, ?x2) \wedge di(?x2, ?x3) \implies ppi(?x1, ?x3)$	SubPropertyChain
108	$mmi(?x1, ?x2) \wedge si(?x2, ?x3) \implies m(?x1, ?x3)$	SubPropertyChain
109	$mmi(?x1, ?x2) \wedge oi(?x2, ?x3) \implies o(?x1, ?x3)$	SubPropertyChain
110	$mmi(?x1, ?x2) \wedge moi(?x2, ?x3) \implies s(?x1, ?x3)$	SubPropertyChain
111	$mmi(?x1, ?x2) \wedge ooi(?x2, ?x3) \implies d(?x1, ?x3)$	SubPropertyChain
112	$mmi(?x1, ?x2) \wedge mi(?x2, ?x3) \implies fi(?x1, ?x3)$	SubPropertyChain
113	$mmi(?x1, ?x2) \wedge mmi(?x2, ?x3) \implies eq(?x1, ?x3)$	SubPropertyChain
114	$mmi(?x1, ?x2) \wedge omi(?x2, ?x3) \implies f(?x1, ?x3)$	SubPropertyChain
115	$mmi(?x1, ?x2) \wedge ppi(?x2, ?x3) \implies di(?x1, ?x3)$	SubPropertyChain
116	$mmi(?x1, ?x2) \wedge m(?x2, ?x3) \implies si(?x1, ?x3)$	SubPropertyChain
117	$mmi(?x1, ?x2) \implies mmi(?x2, ?x1)$	SymmetricProperty
118	$ooi(?x1, ?x2) \wedge eq(?x2, ?x3) \implies ooi(?x1, ?x3)$	SubPropertyChain
119	$ooi(?x1, ?x2) \wedge s(?x2, ?x3) \implies ooi(?x1, ?x3)$	SubPropertyChain
120	$ooi(?x1, ?x2) \wedge d(?x2, ?x3) \implies ooi(?x1, ?x3)$	SubPropertyChain
121	$ooi(?x1, ?x2) \wedge f(?x2, ?x3) \implies ooi(?x1, ?x3)$	SubPropertyChain
122	$ooi(?x1, ?x2) \wedge mi(?x2, ?x3) \implies di(?x1, ?x3)$	SubPropertyChain
123	$ooi(?x1, ?x2) \wedge mmi(?x2, ?x3) \implies di(?x1, ?x3)$	SubPropertyChain
124	$ooi(?x1, ?x2) \wedge ppi(?x2, ?x3) \implies di(?x1, ?x3)$	SubPropertyChain
125	$ooi(?x1, ?x2) \wedge m(?x2, ?x3) \implies di(?x1, ?x3)$	SubPropertyChain
126	$ooi(?x1, ?x2) \implies ooi(?x2, ?x1)$	SymmetricProperty
127	$ppi(?x1, ?x2) \wedge eq(?x2, ?x3) \implies ppi(?x1, ?x3)$	SubPropertyChain
128	$ppi(?x1, ?x2) \wedge fi(?x2, ?x3) \implies ppi(?x1, ?x3)$	SubPropertyChain
129	$ppi(?x1, ?x2) \wedge di(?x2, ?x3) \implies ppi(?x1, ?x3)$	SubPropertyChain
130	$ppi(?x1, ?x2) \wedge si(?x2, ?x3) \implies ppi(?x1, ?x3)$	SubPropertyChain
131	$ppi(?x1, ?x2) \wedge moi(?x2, ?x3) \implies d(?x1, ?x3)$	SubPropertyChain
132	$ppi(?x1, ?x2) \wedge ooi(?x2, ?x3) \implies d(?x1, ?x3)$	SubPropertyChain
133	$ppi(?x1, ?x2) \wedge mmi(?x2, ?x3) \implies d(?x1, ?x3)$	SubPropertyChain
134	$ppi(?x1, ?x2) \wedge omi(?x2, ?x3) \implies d(?x1, ?x3)$	SubPropertyChain
135	$ppi(?x1, ?x2) \implies ppi(?x2, ?x1)$	SymmetricProperty
136	$omi(?x1, ?x2) \wedge eq(?x2, ?x3) \implies omi(?x1, ?x3)$	SubPropertyChain
137	$omi(?x1, ?x2) \wedge s(?x2, ?x3) \implies ooi(?x1, ?x3)$	SubPropertyChain
138	$omi(?x1, ?x2) \wedge d(?x2, ?x3) \implies ooi(?x1, ?x3)$	SubPropertyChain
139	$omi(?x1, ?x2) \wedge f(?x2, ?x3) \implies omi(?x1, ?x3)$	SubPropertyChain
140	$omi(?x1, ?x2) \wedge mi(?x2, ?x3) \implies di(?x1, ?x3)$	SubPropertyChain
141	$omi(?x1, ?x2) \wedge mmi(?x2, ?x3) \implies si(?x1, ?x3)$	SubPropertyChain
142	$omi(?x1, ?x2) \wedge ppi(?x2, ?x3) \implies di(?x1, ?x3)$	SubPropertyChain

143	$\text{omi}(?x1, ?x2) \wedge \text{m}(?x2, ?x3) \Longrightarrow \text{si}(?x1, ?x3)$	SubPropertyChain
144	$\text{omi}(?x1, ?x2) \Longrightarrow \text{moi}(?x2, ?x1)$	InverseProperty
145	$\text{moi}(?x1, ?x2) \wedge \text{eq}(?x2, ?x3) \Longrightarrow \text{moi}(?x1, ?x3)$	SubPropertyChain
146	$\text{moi}(?x1, ?x2) \wedge \text{s}(?x2, ?x3) \Longrightarrow \text{moi}(?x1, ?x3)$	SubPropertyChain
147	$\text{moi}(?x1, ?x2) \wedge \text{d}(?x2, ?x3) \Longrightarrow \text{ooi}(?x1, ?x3)$	SubPropertyChain
148	$\text{moi}(?x1, ?x2) \wedge \text{f}(?x2, ?x3) \Longrightarrow \text{ooi}(?x1, ?x3)$	SubPropertyChain
149	$\text{moi}(?x1, ?x2) \wedge \text{mi}(?x2, ?x3) \Longrightarrow \text{fi}(?x1, ?x3)$	SubPropertyChain
150	$\text{moi}(?x1, ?x2) \wedge \text{mmi}(?x2, ?x3) \Longrightarrow \text{fi}(?x1, ?x3)$	SubPropertyChain
151	$\text{moi}(?x1, ?x2) \wedge \text{ppi}(?x2, ?x3) \Longrightarrow \text{di}(?x1, ?x3)$	SubPropertyChain
152	$\text{moi}(?x1, ?x2) \wedge \text{m}(?x2, ?x3) \Longrightarrow \text{di}(?x1, ?x3)$	SubPropertyChain
153	$\text{moi}(?x1, ?x2) \Longrightarrow \text{omi}(?x2, ?x1)$	InverseProperty
154	$\text{in}(?x1, ?x2) \wedge \text{eq}(?x2, ?x3) \Longrightarrow \text{in}(?x1, ?x3)$	SubPropertyChain
155	$\text{eq}(?x1, ?x2) \Longrightarrow \text{in}(?x1, ?x2)$	SubProperty
156	$\text{s}(?x1, ?x2) \Longrightarrow \text{in}(?x1, ?x2)$	SubProperty
157	$\text{d}(?x1, ?x2) \Longrightarrow \text{in}(?x1, ?x2)$	SubProperty
158	$\text{f}(?x1, ?x2) \Longrightarrow \text{in}(?x1, ?x2)$	SubProperty
159	$\text{dis}(?x1, ?x2) \wedge \text{eq}(?x2, ?x3) \Longrightarrow \text{in}(?x1, ?x3)$	SubPropertyChain
160	$\text{mmi}(?x1, ?x2) \Longrightarrow \text{dis}(?x1, ?x2)$	SubProperty
161	$\text{m}(?x1, ?x2) \Longrightarrow \text{dis}(?x1, ?x2)$	SubProperty
162	$\text{mi}(?x1, ?x2) \Longrightarrow \text{dis}(?x1, ?x2)$	SubProperty
163	$\text{ppi}(?x1, ?x2) \Longrightarrow \text{dis}(?x1, ?x2)$	SubProperty
164	$\text{gT}(?x1, ?x2) \Longrightarrow \text{lT}(?x2, ?x1)$	InverseProperty
165	$\text{lT}(?x1, ?x2) \Longrightarrow \text{gT}(?x2, ?x1)$	InverseProperty
166	$\text{isEq}(?x1, ?x2) \Longrightarrow \text{isEq}(?x2, ?x1)$	SymmetricProperty
167	$\text{gT}(?x1, ?x2) \wedge \text{eq}(?x2, ?x3) \Longrightarrow \text{gT}(?x1, ?x3)$	SubPropertyChain
168	$\text{lT}(?x1, ?x2) \wedge \text{eq}(?x2, ?x3) \Longrightarrow \text{lT}(?x1, ?x3)$	SubPropertyChain
169	$\text{isEq}(?x1, ?x2) \wedge \text{eq}(?x2, ?x3) \Longrightarrow \text{isEq}(?x1, ?x3)$	SubPropertyChain
170	$\text{s}(?x1, ?x2) \Longrightarrow \text{lT}(?x1, ?x2)$	SubProperty
171	$\text{d}(?x1, ?x2) \Longrightarrow \text{lT}(?x1, ?x2)$	SubProperty
172	$\text{f}(?x1, ?x2) \Longrightarrow \text{lT}(?x1, ?x2)$	SubProperty
173	$\text{lT}(?x1, ?x2) \wedge \text{lT}(?x2, ?x3) \Longrightarrow \text{lT}(?x1, ?x3)$	TransitiveProperty
174	$\text{lT}(?x1, ?x2) \wedge \text{isEq}(?x2, ?x3) \Longrightarrow \text{lT}(?x1, ?x3)$	SubPropertyChain
175	$\text{gT}(?x1, ?x2) \wedge \text{gT}(?x2, ?x3) \Longrightarrow \text{gT}(?x1, ?x3)$	TransitiveProperty
176	$\text{gT}(?x1, ?x2) \wedge \text{isEq}(?x2, ?x3) \Longrightarrow \text{gT}(?x1, ?x3)$	SubPropertyChain
177	$\text{isEq}(?x1, ?x2) \wedge \text{isEq}(?x2, ?x3) \Longrightarrow \text{isEq}(?x1, ?x3)$	TransitiveProperty
178	$\text{isEq}(?x1, ?x2) \wedge \text{lT}(?x2, ?x3) \Longrightarrow \text{lT}(?x1, ?x3)$	SubPropertyChain
179	$\text{isEq}(?x1, ?x2) \wedge \text{gT}(?x2, ?x3) \Longrightarrow \text{gT}(?x1, ?x3)$	SubPropertyChain

Le tableau suivant donne les 80 règles d'inférence sur les assertions spatiales selon la syntaxe HRS [Hor+04] suivies de leur type OWL 2 :

n^o	rule	type
1	EQ(?x1, ?x2) $\wedge$ EQ(?x2, ?x3) $\Longrightarrow$ EQ(?x1, ?x3)	TransitiveProperty
2	EQ(?x1, ?x2) $\wedge$ DC(?x2, ?x3) $\Longrightarrow$ DC(?x1, ?x3)	SubPropertyChain
3	EQ(?x1, ?x2) $\wedge$ EC(?x2, ?x3) $\Longrightarrow$ EC(?x1, ?x3)	SubPropertyChain
4	EQ(?x1, ?x2) $\wedge$ PO(?x2, ?x3) $\Longrightarrow$ PO(?x1, ?x3)	SubPropertyChain
5	EQ(?x1, ?x2) $\wedge$ TPP(?x2, ?x3) $\Longrightarrow$ TPP(?x1, ?x3)	SubPropertyChain
6	EQ(?x1, ?x2) $\wedge$ NTPP(?x2, ?x3) $\Longrightarrow$ NTPP(?x1, ?x3)	SubPropertyChain
7	EQ(?x1, ?x2) $\wedge$ TPPi(?x2, ?x3) $\Longrightarrow$ TPPi(?x1, ?x3)	SubPropertyChain
8	EQ(?x1, ?x2) $\wedge$ NTPPi(?x2, ?x3) $\Longrightarrow$ NTPPi(?x1, ?x3)	SubPropertyChain
9	EQ(?x1, ?x2) $\Longrightarrow$ EQ(?x2, ?x1)	SymmetricProperty
10	DC(?x1, ?x2) $\wedge$ EQ(?x2, ?x3) $\Longrightarrow$ DC(?x1, ?x3)	SubPropertyChain
11	DC(?x1, ?x2) $\wedge$ TPPi(?x2, ?x3) $\Longrightarrow$ DC(?x1, ?x3)	SubPropertyChain
12	DC(?x1, ?x2) $\wedge$ NTPPi(?x2, ?x3) $\Longrightarrow$ DC(?x1, ?x3)	SubPropertyChain
13	DC(?x1, ?x2) $\Longrightarrow$ DC(?x2, ?x1)	SymmetricProperty
14	EC(?x1, ?x2) $\wedge$ EQ(?x2, ?x3) $\Longrightarrow$ EC(?x1, ?x3)	SubPropertyChain
15	EC(?x1, ?x2) $\wedge$ NTPPi(?x2, ?x3) $\Longrightarrow$ DC(?x1, ?x3)	SubPropertyChain
16	EC(?x1, ?x2) $\Longrightarrow$ EC(?x2, ?x1)	SymmetricProperty
17	PO(?x1, ?x2) $\wedge$ EQ(?x2, ?x3) $\Longrightarrow$ PO(?x1, ?x3)	SubPropertyChain
18	PO(?x1, ?x2) $\Longrightarrow$ PO(?x2, ?x1)	SymmetricProperty
19	TPP(?x1, ?x2) $\wedge$ EQ(?x2, ?x3) $\Longrightarrow$ TPP(?x1, ?x3)	SubPropertyChain
20	TPP(?x1, ?x2) $\wedge$ DC(?x2, ?x3) $\Longrightarrow$ DC(?x1, ?x3)	SubPropertyChain
21	TPP(?x1, ?x2) $\wedge$ NTPP(?x2, ?x3) $\Longrightarrow$ NTPP(?x1, ?x3)	SubPropertyChain
22	TPP(?x1, ?x2) $\Longrightarrow$ TPPi(?x2, ?x1)	InverseProperty
23	NTPP(?x1, ?x2) $\wedge$ EQ(?x2, ?x3) $\Longrightarrow$ NTPP(?x1, ?x3)	SubPropertyChain
24	NTPP(?x1, ?x2) $\wedge$ DC(?x2, ?x3) $\Longrightarrow$ DC(?x1, ?x3)	SubPropertyChain
25	NTPP(?x1, ?x2) $\wedge$ EC(?x2, ?x3) $\Longrightarrow$ DC(?x1, ?x3)	SubPropertyChain
26	NTPP(?x1, ?x2) $\wedge$ TPP(?x2, ?x3) $\Longrightarrow$ NTPP(?x1, ?x3)	SubPropertyChain
27	NTPP(?x1, ?x2) $\wedge$ NTPP(?x2, ?x3) $\Longrightarrow$ NTPP(?x1, ?x3)	TransitiveProperty
28	NTPP(?x1, ?x2) $\Longrightarrow$ NTPPi(?x2, ?x1)	InverseProperty
29	TPPi(?x1, ?x2) $\wedge$ EQ(?x2, ?x3) $\Longrightarrow$ TPPi(?x1, ?x3)	SubPropertyChain
30	TPPi(?x1, ?x2) $\wedge$ NTPPi(?x2, ?x3) $\Longrightarrow$ NTPPi(?x1, ?x3)	SubPropertyChain
31	TPPi(?x1, ?x2) $\Longrightarrow$ TPP(?x2, ?x1)	InverseProperty
32	NTPPi(?x1, ?x2) $\wedge$ EQ(?x2, ?x3) $\Longrightarrow$ NTPPi(?x1, ?x3)	SubPropertyChain
33	NTPPi(?x1, ?x2) $\wedge$ TPPI(?x2, ?x3) $\Longrightarrow$ NTPPi(?x1, ?x3)	SubPropertyChain
34	NTPPi(?x1, ?x2) $\wedge$ NTPPi(?x2, ?x3) $\Longrightarrow$ NTPPi(?x1, ?x3)	TransitiveProperty
35	NTPPi(?x1, ?x2) $\Longrightarrow$ NTPP(?x2, ?x1)	InverseProperty
36	vc(?x1, ?x2) $\Longrightarrow$ vc(?x2, ?x1)	SymmetricProperty
37	c(?x1, ?x2) $\Longrightarrow$ c(?x2, ?x1)	SymmetricProperty
38	f(?x1, ?x2) $\Longrightarrow$ f(?x2, ?x1)	SymmetricProperty

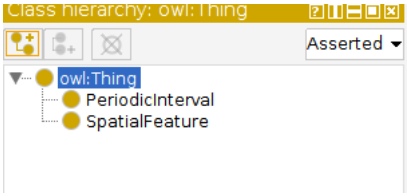
39	$vf(?x1, ?x2) \implies vf(?x2, ?x1)$	SymmetricProperty
40	$N(?x1, ?x2) \implies S(?x2, ?x1)$	InverseProperty
41	$S(?x1, ?x2) \implies N(?x2, ?x1)$	InverseProperty
42	$E(?x1, ?x2) \implies W(?x2, ?x1)$	InverseProperty
43	$W(?x1, ?x2) \implies E(?x2, ?x1)$	InverseProperty
44	$SE(?x1, ?x2) \implies NW(?x2, ?x1)$	InverseProperty
45	$NW(?x1, ?x2) \implies SE(?x2, ?x1)$	InverseProperty
46	$SO(?x1, ?x2) \implies NE(?x2, ?x1)$	InverseProperty
47	$NE(?x1, ?x2) \implies SO(?x2, ?x1)$	InverseProperty
48	$N(?x1, ?x2) \wedge N(?x2, ?x3) \implies N(?x1, ?x3)$	TransitiveProperty
49	$S(?x1, ?x2) \wedge S(?x2, ?x3) \implies S(?x1, ?x3)$	TransitiveProperty
50	$E(?x1, ?x2) \wedge E(?x2, ?x3) \implies E(?x1, ?x3)$	TransitiveProperty
51	$W(?x1, ?x2) \wedge W(?x2, ?x3) \implies W(?x1, ?x3)$	TransitiveProperty
52	$SE(?x1, ?x2) \wedge SE(?x2, ?x3) \implies SE(?x1, ?x3)$	TransitiveProperty
53	$SO(?x1, ?x2) \wedge SO(?x2, ?x3) \implies SO(?x1, ?x3)$	TransitiveProperty
54	$NE(?x1, ?x2) \wedge NE(?x2, ?x3) \implies NE(?x1, ?x3)$	TransitiveProperty
55	$NO(?x1, ?x2) \wedge NO(?x2, ?x3) \implies NO(?x1, ?x3)$	TransitiveProperty
56	$hh(?x1, ?x2) \implies hh(?x2, ?x1)$	SymmetricProperty
57	$il(?x1, ?x2) \implies ih(?x2, ?x1)$	InverseProperty
58	$ih(?x1, ?x2) \implies il(?x2, ?x1)$	InverseProperty
59	$hh(?x1, ?x2) \wedge hh(?x2, ?x3) \implies hh(?x1, ?x3)$	TransitiveProperty
60	$hh(?x1, ?x2) \wedge il(?x2, ?x3) \implies il(?x1, ?x3)$	SubPropertyChain
61	$hh(?x1, ?x2) \wedge ih(?x2, ?x3) \implies ih(?x1, ?x3)$	SubPropertyChain
62	$il(?x1, ?x2) \wedge hh(?x2, ?x3) \implies il(?x1, ?x3)$	SubPropertyChain
63	$il(?x1, ?x2) \wedge il(?x2, ?x3) \implies il(?x1, ?x3)$	TransitiveProperty
64	$ih(?x1, ?x2) \wedge hh(?x2, ?x3) \implies ih(?x1, ?x3)$	SubPropertyChain
65	$ih(?x1, ?x2) \wedge ih(?x2, ?x3) \implies ih(?x1, ?x3)$	TransitiveProperty
66	$isEQs(?x1, ?x2) \implies isEQs(?x2, ?x1)$	SymmetricProperty
67	$iB(?x1, ?x2) \implies iS(?x2, ?x1)$	InverseProperty
68	$iS(?x1, ?x2) \implies iB(?x2, ?x1)$	InverseProperty
69	$isEqs(?x1, ?x2) \wedge isEqs(?x2, ?x3) \implies isEqs(?x1, ?x3)$	TransitiveProperty
70	$isEqs(?x1, ?x2) \wedge iS(?x2, ?x3) \implies iS(?x1, ?x3)$	SubPropertyChain
71	$isEqs(?x1, ?x2) \wedge iB(?x2, ?x3) \implies iB(?x1, ?x3)$	SubPropertyChain
72	$iS(?x1, ?x2) \wedge isEqs(?x2, ?x3) \implies iS(?x1, ?x3)$	SubPropertyChain
73	$iS(?x1, ?x2) \wedge iS(?x2, ?x3) \implies iS(?x1, ?x3)$	TransitiveProperty
74	$iB(?x1, ?x2) \wedge isEQs(?x2, ?x3) \implies iB(?x1, ?x3)$	SubPropertyChain
75	$iB(?x1, ?x2) \wedge iB(?x2, ?x3) \implies iB(?x1, ?x3)$	TransitiveProperty
76	$EQ(?x1, ?x2) \implies isEQs(?x1, ?x2)$	SubProperty
77	$TPP(?x1, ?x2) \implies iS(?x1, ?x2)$	SubProperty

78	NTPP(?x1, ?x2) $\Longrightarrow$ iS(?x1, ?x2)	SubProperty
79	TPPi(?x1, ?x2) $\Longrightarrow$ iB(?x1, ?x2)	SubProperty
80	NTPPi(?x1, ?x2) $\Longrightarrow$ iB(?x1, ?x2)	SubProperty

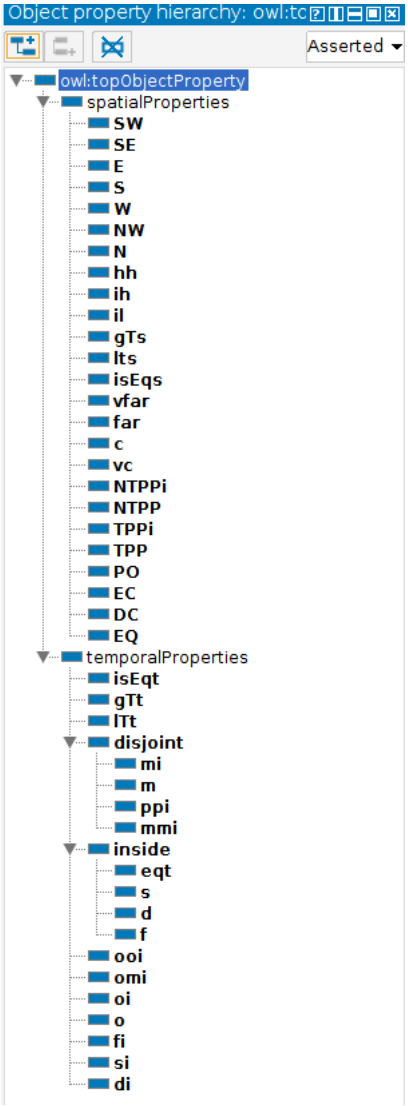
Ontologie

L'ontologie est disponible dans le même dépot que le code source du logiciel VSPCC : `https://sourcesup.renater.fr/projects/vspcc`

Nous donnons ici un aperçu de sa constitution au travers des figures 5.12a et 5.12b qui montrent respectivement les classes et les propriétés de l'ontologie.



(a) Classes de l'ontologie.



(b) Propriétés de l'ontologie.

BIBLIOGRAPHIE

[Agu02] Jose AGUILAR, « Adaptive random fuzzy cognitive maps », in : *Ibero-American Conference on Artificial Intelligence*, Springer, 2002, p. 402-410.

[All96] Florence ALLARD-POESI, « Cartes cognitives : pour ne pas jeter le bébé avec l'eau du bain », in : *Conférence Internationale de Management Stratégique*, 1996.

[All83] James F. ALLEN, « Maintaining Knowledge About Temporal Intervals », in : *ACM* 26.*11* (nov. 1983), p. 832-843, ISSN : 0001-0782, DOI : 10.1145/182.358434, URL : http://doi.acm.org/10.1145/182.358434.

[All86] Lloyd ALLISON, *A practical introduction to denotational semantics*, t. 23, Cambridge University Press, 1986.

[And+00] Ion ANDROUTSOPOULOS, John KOUTSIAS, Konstantinos V CHANDRINOS, George PALIOURAS et Constantine D SPYROPOULOS, « An evaluation of naive bayesian anti-spam filtering », in : *arXiv preprint cs/0006013* (2000).

[Ang18] Renzo ANGLES, « The Property Graph Database Model. », in : *Proceedings of the 12th Alberto Mendelzon International Workshop on Foundations of Data Management*, 2018.

[Ang+18] Renzo ANGLES et al., « G-CORE : A core for future graph query languages », in : *Proceedings of the 2018 International Conference on Management of Data*, 2018, p. 1421-1432.

[Ang+17] Renzo ANGLES, Marcelo ARENAS, Pablo BARCELÓ, Aidan HOGAN, Juan REUTTER et Domagoj VRGO, « Foundations of modern query languages for graph databases », in : *ACM Computing Surveys (CSUR)* 50.*5* (2017), p. 1-40.

[AG08] Renzo ANGLES et Claudio GUTIERREZ, « The expressive power of SPARQL », in : *International Semantic Web Conference*, Springer, 2008, p. 114-129.

[AV04] Grigoris ANTONIOU et Frank VAN HARMELEN, *A semantic web primer*, MIT press, 2004.

[AD93] Paolo ATZENI et Valeria DE ANTONELLIS, *Relational database theory*, Benjamin/-Cummings Redwood City, CA, 1993.

[Axe15] Robert AXELROD, *Structure of decision : The cognitive maps of political elites*, Princeton university press, 2015.

[Baa+03] Franz BAADER, Diego CALVANESE, Deborah MCGUINNESS, Peter PATEL SCHNEIDER, Daniele NARDI et al., *The description logic handbook : Theory, implementation and applications*, Cambridge university press, 2003.

[BO00] Philippe BALBIANI et Aomar OSMANI, « A model for reasoning about topologic relations between cyclic intervals », in : *Principles of Knowledge Representation and Reasoning*, 2000, p. 378-385.

[BPL03] Josianne BASQUE, Béatrice PUDELKO et Denis LEGROS, « Une expérience de construction de cartes conceptuelles dans un contexte de téléapprentissage universitaire », in : *Environnements Informatiques pour l'Apprentissage Humain*, 2003.

[BK11] Robert BATTLE et Dave KOLAS, « Geosparql : enabling a geospatial semantic web », in : *Semantic Web Journal* 3.*4* (2011), p. 355-370.

[BHL01] Tim BERNERS-LEE, James A HENDLER et Ora LASSILA, *The Semantic Web Scientific American, 284 (5) : 34–43*, 2001.

[BWB77] Michel BOUGON, Karl WEICK et Din BINKHORST, « Cognition in organizations : An analysis of the Utrecht Jazz Orchestra », in : *Administrative Science Quarterly* (1977), p. 606-639.

[Bra79] Ronald J BRACHMAN, « On the epistemological status of semantic networks », in : *Associative networks*, Elsevier, 1979, p. 3-50.

[Bra+00] Tim BRAY, Jean PAOLI, C Michael SPERBERG-MCQUEEN, Eve MALER, Franois YERGEAU et al., *Extensible markup language (XML) 1.0*, 2000.

[Bus+06] Justin Eliot BUSCH, Albert Deirchow LIN, Patrick John GRAYDON et Maureen CAUDILL, *Ontology-based parser for natural language processing*, US Patent 7,027,974, avr. 2006.

[BB93] Tony BUZAN et Barry BUZAN, « The mind map book how to use radiant thinking to maximise your brain's untapped potential », in : *New York : Plume* (1993).

[BB06] Tony BUZAN et Barry BUZAN, *The mind map book*, Pearson Education, 2006.

[CSG04] Rafael CANO, Carmen SORDO et José M GUTIÉRREZ, « Applications of Bayesian networks in meteorology », in : *Advances in Bayesian networks*, Springer, 2004, p. 309-328.

[CT99] João Paulo CARVALHO et Jose AB TOMÈ, « Rule based fuzzy cognitive maps-fuzzy causal relations », in : *Computational Intelligence for Modelling, Control and Automation, Edited by M. Mohammadian* 199.*9* (1999).

[CT01] João Paulo CARVALHO et José Alberto Batista TOME, « Rule based fuzzy cognitive maps-expressing time in qualitative system dynamics », in : *10th IEEE International Conference on Fuzzy Systems.(Cat. No. 01CH37297)*, t. 1, IEEE, 2001, p. 280-283.

[CT+95] Joseph CATANZANO, Olivier THÉBAUD et al., *Le littoral : pour une approche de la regulation des conflicts d'usage*, Paris (France) Inst. oceanographique/IFREMER, 1995.

[Cha+76] Donald D. CHAMBERLIN, Morton M. ASTRAHAN, Kapali P. ESWARAN, Patricia P. GRIFFITHS, Raymond A. LORIE, James W. MEHL, Phyllis REISNER et Bradford W. WADE, « SEQUEL 2 : a unified approach to data definition, manipulation, and control », in : *IBM Journal of Research and Development* 20.*6* (1976), p. 560-575.

[Cha10] Lionel CHAUVIN, « Models of cognitive maps extended to the notions of context and scale », Theses, Université d'Angers, sept. 2010, URL : `https://tel.archives-ouvertes.fr/tel-00585259`.

[CGL08] Lionel CHAUVIN, David GENEST et Stéphane LOISEAU, « Les cartes cognitives hiérarchiques. », in : *EGC*, 2008, p. 91.

[CGL09] Lionel CHAUVIN, David GENEST et Stéphane LOISEAU, « Ontological cognitive map », in : *International Journal on Artificial Intelligence Tools* 18.*05* (2009), p. 697-716.

[Chr11] Gwen CHRISTIANSEN, « Modélisation du savoir porté par les acteurs dun système : application aux pêcheurs à la coquille en rade de Brest », in : *Mémoire de Master, Agrocampus Ouest, Saint-Brieuc, France* 58 (2011), p. 95-98.

[CFT08] Kendall Grant CLARK, Lee FEIGENBAUM et Elias TORRES, « SPARQL protocol for RDF », in : *World Wide Web Consortium (W3C) Recommendation* 86 (2008).

[Coh+97] Anthony G COHN, Brandon BENNETT, John GOODAY et Nicholas Mark GOTTS, « Qualitative spatial representation and reasoning with the region connection calculus », in : *GeoInformatica* 1.*3* (1997), p. 275-316.

[CFG03] Oscar CORCHO, Mariano FERNÁNDEZ-LÓPEZ et Asunción GÓMEZ-PÉREZ, « Methodologies, tools and languages for building ontologies. Where is their meeting point ? », in : *Data & knowledge engineering* 46.*1* (2003), p. 41-64.

[Cos08] Pierre COSSETTE, « La cartographie cognitive vue d'une perspective subjectiviste : mise à l'épreuve d'une nouvelle approche », in : *M@ n@ gement* 11.*3* (2008), p. 259-281.

[CA92] Pierre COSSETTE et Michel AUDET, « Mapping of an idiosyncratic schema », in : *Journal of Management Studies* 29.*3* (1992), p. 325-347.

[DMN09] Antonio DE NICOLA, Michele MISSIKOFF et Roberto NAVIGLI, « A software engineering approach to ontology building », in : *Information systems* 34.*2* (2009), p. 258-275.

[DLN06] Erick DELAGE, Honglak LEE et Andrew Y NG, « A dynamic bayesian network model for autonomous 3d reconstruction from a single indoor image », in : *2006 IEEE computer society conference on computer vision and pattern recognition (CVPR'06)*, t. 2, IEEE, 2006, p. 2418-2428.

[Dev+91] Prem DEVANBU, Ron BRACHMAN, Peter G SELFRIDGE et Bruce W BALLARD, « LaSSIE : A knowledge-based software information system », in : *Communications of the ACM* 34.*5* (1991), p. 34-49.

[DSS12] Jacob DIJKSTRA, Timo SEPTER et Frans STOKMAN, « Detecting and Measuring Crucial Differences between Cognitive Maps », in : *Rationality and Society* 24 (nov. 2012), p. 383-407, DOI : 10.1177/1043463112463915.

[DBS05] Arta DILO, Rolf de BY et Alfred STEIN, « A proposal for spatial relations between vague objects », in : août 2005.

[Don08] Tiansi DONG, « A comment on RCC : from RCC to RCC++ », in : *Journal of Philosophical Logic* 37.*4* (2008), p. 319-352.

[DF99] James K DOYLE et David N FORD, « Mental models concepts revisited : some clarifications and a reply to Lane », in : *System Dynamics Review : The Journal of the System Dynamics Society* 15.*4* (1999), p. 411-415.

[Doy79] Jon DOYLE, « A truth maintenance system », in : *Artificial Intelligence* 12.*3* (1979), p. 231-272, ISSN : 0004-3702, DOI : https://doi.org/10.1016/0004-3702(79)90008-0, URL : http://www.sciencedirect.com/science/article/pii/0004370279900080.

[Dup01] Dominique DUPILET, « Le règlement des conflits dusage dans la zone côtière entre pêche professionnelle et autres activités », in : *Paris : Assemblée nationale, Rapport au Premier ministre* (2001).

[Ede92] Colin EDEN, « On the nature of cognitive maps », in : *Journal of management studies* 29.*3* (1992), p. 261-265.

[Ede04] Colin EDEN, « Analyzing cognitive maps to help structure issues or problems », in : *European Journal of Operational Research* 159.*3* (2004), p. 673-686, ISSN : 0377-2217, DOI : https://doi.org/10.1016/S0377-2217(03)00431-4, URL : http://www.sciencedirect.com/science/article/pii/S0377221703004314.

[EA13] Colin EDEN et Fran ACKERMANN, *Making strategy : The journey of strategic management*, Sage, 2013.

[Ege89] Max J EGENHOFER, « A formal definition of binary topological relationships », in : *International conference on foundations of data organization and algorithms*, Springer, 1989, p. 457-472.

[Ege91] Max J EGENHOFER, « Reasoning about binary topological relations », in : *Symposium on Spatial Databases*, Springer, 1991, p. 141-160.

[EF91] Max J EGENHOFER et Robert D FRANZOSA, « Point-set topological spatial relations », in : *International Journal of Geographical Information System* 5.*2* (1991), p. 161-174.

[Erm+14] Vadim ERMOLAYEV, Sotiris BATSAKIS, Natalya KEBERLE, Olga TATARINTSEVA et Grigoris ANTONIOU, « Ontologies of Time : Review and Trends. », in : *International Journal of Computer Science & Applications* 11.*3* (2014), p. 57-115.

[Ea08] ERMOLAYEV et AL, « Fuzzy time intervals for simulating actions », in : *International United Information Systems Conference*, Springer, 2008, p. 429-444.

[For95] KENNETH FORBUS, *Qualitative spatial reasoning : Framework and frontiers*, 1995.

[FNF90] Kenneth D FORBUS, Paul NIELSEN et Boi FALTINGS, « Qualitative kinematics : A framework », in : *Readings in qualitative reasoning about physical systems*, Elsevier, 1990, p. 562-567.

[Fra+18a] Nadime FRANCIS et al., « Formal semantics of the language cypher », in : *arXiv preprint arXiv :1802.09984* (2018).

[Fra+18b] Nadime FRANCIS et al., « Cypher : An evolving query language for property graphs », in : *Proceedings of the 2018 International Conference on Management of Data*, 2018, p. 1433-1445.

[Fra94] Enrico FRANCONI, *Description logics for natural language processing*, Istituto per la Ricerca Scientifica e Tecnologica, 1994.

[Fra92] Andrew U FRANK, « Qualitative spatial reasoning about distances and directions in geographic space », in : *Journal of Visual Languages & Computing* 3.*4* (1992), p. 343-371.

[Fra96] Andrew U FRANK, « Qualitative spatial reasoning : Cardinal directions as an example », in : *International Journal of Geographical Information Science* 10.*3* (1996), p. 269-290.

[FG99] Nir FRIEDMAN et Moises GOLDSZMIDT, *Learning Bayesian networks from data*, Morgan Kaufmann, 1999.

[GOR09] Matteo GAETA, Francesco ORCIUOLI et Pierluigi RITROVATO, « Advanced ontology management system for personalised e-Learning », in : *Knowledge-Based Systems* 22.*4* (2009), p. 292-301.

[Gal09] Antony GALTON, « Spatial and temporal knowledge representation », in : *Earth Science Informatics* 2.*3* (2009), p. 169-187.

[GR05] Françoise GOURMELON et Marc ROBIN, *SIG et littoral. Traité IGAT (Information Géographique et Aménagement du Territoire)*, 2005.

[GN84] D Bob GOWIN et Joseph D NOVAK, « Learning how to learn », in : *USA : Cambridge University* (1984).

[GB07] Rolf GRÜTTER et Bettina BAUER-MESSMER, « Combining OWL with RCC for Spatioterminological Reasoning on Environmental Data. », in : *OWLED*, 2007.

[Hag92] Masafumi HAGIWARA, « Extended fuzzy cognitive maps », in : *[1992 Proceedings] IEEE International Conference on Fuzzy Systems*, IEEE, 1992, p. 795-801.

[Har17] Mounira HARZALLAH, « Contributions à l'Ingénierie des Connaissances : Construction et Validation dOntologie et Mesures Sémantique », Habilitation à Diriger des Recherches (HDR), 2017.

[HG02] Eveline M HELSPER et Linda C GAAG, « Building Bayesian networks through ontologies », in : *Proceedings of the 15th European Conference on Artificial Intelligence*, IOS Press, 2002, p. 680-684.

[HDN04] Nicola HENZE, Peter DOLOG et Wolfgang NEJDL, « Reasoning and ontologies for personalized e-learning in the semantic web », in : *Journal of Educational Technology & Society* 7.*4* (2004), p. 82-97.

[HMM00] Ivan HERMAN, Guy MELANÇON et M Scott MARSHALL, « Graph visualization and navigation in information visualization : A survey », in : *IEEE Transactions on visualization and computer graphics* 6.*1* (2000), p. 24-43.

[Hit+09] Pascal HITZLER, Markus KRÖTZSCH, Bijan PARSIA, Peter F PATEL-SCHNEIDER, Sebastian RUDOLPH et al., « OWL 2 web ontology language primer », in : *W3C recommendation* 27.*1* (2009), p. 123.

[Hob+02] Jerry R HOBBS, George FERGUSON, James ALLEN, P HAYES, I NILES et A PEASE, *A daml ontology of time*, 2002.

[HP06] Jerry R HOBBS et Feng PAN, « Time ontology in OWL », in : *W3C working draft* 27 (2006), p. 133.

[Hog+10] Frederik HOGENBOOM, Bram BORGMAN, Flavius FRASINCAR et Uzay KAYMAK, « Spatial knowledge representation on the semantic web », in : *2010 IEEE Fourth International Conference on Semantic Computing*, IEEE, 2010, p. 252-259.

[Hor05a] Ian HORROCKS, « Applications of description logics : State of the art and research challenges », in : *International Conference on Conceptual Structures*, Springer, 2005, p. 78-90.

[HKS06] Ian HORROCKS, Oliver KUTZ et Ulrike SATTLER, « The Even More Irresistible SROIQ. », in : *Kr* 6 (2006), p. 57-67.

[HP03] Ian HORROCKS et Peter F PATEL-SCHNEIDER, « Reducing OWL entailment to description logic satisfiability », in : *International semantic web conference*, Springer, 2003, p. 17-29.

[Hor+05] Ian HORROCKS, Peter F PATEL-SCHNEIDER, Sean BECHHOFER et Dmitry TSARKOV, « OWL rules : A proposal and prototype implementation », in : *Journal of web semantics* 3.*1* (2005), p. 23-40.

[Hor+04] Ian HORROCKS, Peter F PATEL-SCHNEIDER, Harold BOLEY, Said TABET, Benjamin GROSOF, Mike DEAN et al., « SWRL : A semantic web rule language combining OWL and RuleML », in : *W3C Member submission* 21.*79* (2004), p. 1-31.

[HPV03] Ian HORROCKS, Peter F PATEL-SCHNEIDER et Frank VAN HARMELEN, « From SHIQ and RDF to OWL : The making of a web ontology language », in : *Journal of web semantics* 1.*1* (2003), p. 7-26.

[Hor05b] Herman J ter HORST, « Combining RDF and part of OWL with rules : Semantics, decidability, complexity », in : *International Semantic Web Conference*, Springer, 2005, p. 668-684.

[HS03] Bo HU et Nigel SHADBOLT, « Visualising a DL Knowledge Base with DeLogViz. », in : *Description Logics*, 2003.

[JP89] Jeffrey C. JOHNSON et Richard B. POLLNAC, « Introduction to managing marine conflicts », in : *Ocean and Shoreline Management* 12.*3* (1989), Managing Marine Conflicts, p. 191-198, ISSN : 0951-8312, DOI : https://doi.org/10.1016/0951-8312(89)90002-7, URL : http://www.sciencedirect.com/science/article/pii/0951831289900027.

[Kah+97] Charles E KAHN JR, Linda M ROBERTS, Katherine A SHAFFER et Peter HADDAWY, « Construction of a Bayesian network for mammographic diagnosis of breast cancer », in : *Computers in biology and medicine* 27.*1* (1997), p. 19-29.

[Knu64] Donald E KNUTH, « Backus normal form vs. backus naur form », in : *Communications of the ACM* 7.*12* (1964), p. 735-736.

[Kos+86] Bart KOSKO et al., « Fuzzy cognitive maps », in : *International journal of man-machine studies* 24.*1* (1986), p. 65-75.

[Krö+11] Markus KRÖTZSCH, Frederick MAIER, Adila KRISNADHI et Pascal HITZLER, « A better uncle for OWL : Nominal schemas for integrating rules and ontologies », in : *Proceedings of the 20th international conference on World wide web*, 2011, p. 645-654.

[Lad86] LADKIN, « Time Representation :A Taxonomy of Internal Relations. », in : *AAAI*, 1986, p. 360-366.

[Lad87] Peter Bernard LADKIN, « The logic of time representation », thèse de doct., Citeseer, 1987.

[LMD14] Josep Llus LARRIBA-PEY, Norbert MARTNEZ-BAZÁN et David DOMNGUEZ-SAL, « Introduction to graph databases », in : *Reasoning Web International Summer School*, Springer, 2014, p. 171-194.

[Lau92] Steffen L LAURITZEN, « Propagation of probabilities, means, and variances in mixed graphical association models », in : *Journal of the American Statistical Association* 87.*420* (1992), p. 1098-1108.

[LeD13] Aymeric LEDORZE, « Validation, synthesis, and settings of cognitive maps », Theses, Université d'Angers, nov. 2013, URL : https://tel.archives-ouvertes.fr/tel-00956983.

[LeD+12] Aymeric LEDORZE, Lionel CHAUVIN, Laurent GARCIA, David GENEST et Stéphane LOISEAU, « Views and synthesis of cognitive maps », in : *International Conference on Artificial Intelligence : Methodology, Systems, and Applications*, Springer, 2012, p. 119-124.

[LeD+13] Aymeric LEDORZE, Laurent GARCIA, David GENEST et Stéphane LOISEAU, « Validation d'une carte cognitive », in : *13e Conférence Francophone sur lExtraction et la Gestion des Connaissances (EGC2013)*, 2013.

[Leh92] Fritz LEHMANN, « Semantic networks », in : *Computers & Mathematics with Applications* 23.*2-5* (1992), p. 1-50.

[LG89] Douglas B LENAT et Ramanathan V GUHA, *Building large knowledge-based systems ; representation and inference in the Cyc project*, Addison-Wesley Longman Publishing Co., Inc., 1989.

[LY03] Sanjiang LI et Mingsheng YING, « Extensionality of the RCC8 composition table », in : *Fundamenta Informaticae* 55.*3-4* (2003), p. 363-385.

[LMV16] Leonid LIBKIN, Wim MARTENS et Domagoj VRGO, « Querying graphs with data », in : *Journal of the ACM (JACM)* 63.*2* (2016), p. 1-53.

[LSG07] Joshua LIEBERMAN, Raj SINGH et Chris GOAD, « W3c geospatial ontologies », in : *Incubator group report, W3C* (2007).

[Liu+14] Shixia LIU, Weiwei CUI, Yingcai WU et Mengchen LIU, « A survey on information visualization : recent advances and challenges », in : *The Visual Computer* 30.*12* (2014), p. 1373-1393.

[LS99] Zhi-Qiang LIU et Richard SATUR, « Contextual fuzzy cognitive map for decision support in geographic information systems », in : *IEEE Transactions on Fuzzy Systems* 7.*5* (1999), p. 495-507.

[LP82] Georges LOUIS et Alain PIROTTE, « A Denotational Definition of the Semantics of DRC, A Domain Relational Calculus. », in : *VLDB*, 1982, p. 348-356.

[Mag+13] Vijay K MAGO, Hilary K MORDEN, Charles FRITZ, Tiankuang WU, Sara NAMAZI, Parastoo GERANMAYEH, Rakhi CHATTOPADHYAY et Vahid DABBAGHIAN, « Analyzing the impact of social factors on homelessness : a Fuzzy Cognitive Map approach », in : *BMC medical informatics and decision making* 13.*1* (2013), p. 94.

[MMM+04] Frank MANOLA, Eric MILLER, Brian MCBRIDE et al., « RDF primer », in : *W3C recommendation* 10.*1-107* (2004), p. 6.

[Mas61] Margaret MASTERMAN, « Semantic message detection for machine translation, using an interlingua », in : *Proc. 1961 International Conf. on Machine Translation*, 1961, p. 438-475.

[MR03] Daniel D MCCRACKEN et Edwin D REILLY, « Backus-naur form (bnf) », in : (2003).

[MV+04] Deborah L MCGUINNESS, Frank VAN HARMELEN et al., « OWL web ontology language overview », in : *W3C recommendation* 10.*10* (2004), p. 2004.

[MW98] Deborah L MCGUINNESS et Jon R WRIGHT, « An industrial-strength description logic-based configurator platform », in : *IEEE Intelligent Systems and their applications* 13.*4* (1998), p. 69-77.

[Mia+01] Yuan MIAO, Zhi-Qiang LIU, Chee Kheong SIEW et Chun Yan MIAO, « Dynamical cognitive network-an extension of fuzzy cognitive map », in : *IEEE transactions on Fuzzy Systems* 9.*5* (2001), p. 760-770.

[Mih96] George Andrei MIHAILA, *WebSQL : an SQL-like query language for the World Wide Web*, University of Toronto, 1996.

[Min74] Marvin MINSKY, *A Framework for Representing Knowledge*, rapp. tech., USA, 1974.

[Moo06] James MOOR, « The Dartmouth College artificial intelligence conference : The next fifty years », in : *Ai Magazine* 27.*4* (2006), p. 87-87.

[MZ97] Tadeusz MORZY et Maciej ZAKRZEWICZ, « SQL-Like Language for Database Mining. », in : *ADBIS*, 1997, p. 311-317.

[Mos90] Peter D MOSSES, « Denotational semantics », in : *Formal Models and Semantics*, Elsevier, 1990, p. 575-631.

[MC92] Marie-Laure MUGNIER et Michel CHEIN, « Conceptual graphs : Fundamental notions », in : *Revue dintelligence artificielle* 6.*4* (1992), p. 365-406.

[Mur98] Kevin P MURPHY, *Inference and learning in hybrid Bayesian networks*, University of California, Berkeley, Computer Science Division, 1998.

[NFN00] Martin NEIL, Norman FENTON et Lars NIELSON, « Building large-scale Bayesian networks », in : *The Knowledge Engineering Review* 15.*3* (2000), p. 257-284.

[Noh+00] JB NOH, KC LEE, JK KIM, JK LEE et Soung Hie KIM, « A case-based reasoning approach to cognitive map-driven tacitknowledge management », in : *Expert systems with applications* 19.*4* (2000), p. 249-259.

[Nov02] Joseph D NOVAK, « Meaningful learning : The essential factor for conceptual change in limited or inappropriate propositional hierarchies leading to empowerment of learners », in : *Science education* 86.*4* (2002), p. 548-571.

[NC08] Joseph D NOVAK et Alberto J CAÑAS, *The theory underlying concept maps and how to construct and use them*, rapp. tech., 2008.

[ØH07] Peter ØHRSTRØM et Per HASLE, *Temporal logic : from ancient ideas to artificial intelligence*, t. 57, Springer Science & Business Media, 2007.

[Osm99] Aomar OSMANI, « Introduction to Reasoning about Cyclic Intervals », in : *Multiple Approaches to Intelligent Systems*, Springer, 1999, p. 698-706, ISBN : 978-3-540-48765-4.

[ÖÖ03] Uygar ÖZESMI et Stacy ÖZESMI, « A participatory approach to ecosystem conservation : fuzzy cognitive maps and stakeholder group analysis in Uluabat Lake, Turkey », in : *Environmental management* 31.*4* (2003), p. 0518-0531.

[ÖÖ04] Uygar ÖZESMI et Stacy L ÖZESMI, « Ecological models based on peoples knowledge : a multi-step fuzzy cognitive mapping approach », in : *Ecological modelling* 176.*1-2* (2004), p. 43-64.

[PR94] Stephen PALMER et Irvin ROCK, « Rethinking perceptual organization : The role of uniform connectedness », in : *Psychonomic bulletin & review* 1.*1* (1994), p. 29-55.

[PH04] Feng PAN et Jerry R HOBBS, « Time in owl-s », in : *Proceedings of the AAAI Spring Symposium on Semantic Web Services*, 2004, p. 29-36.

[PSG04] EI PAPAGEORGIOU, Chrysostomos D STYLIOS et Peter P GROUMPOS, « Active Hebbian learning algorithm to train fuzzy cognitive maps », in : *International journal of approximate reasoning* 37.*3* (2004), p. 219-249.

[PSG06] Elpiniki I PAPAGEORGIOU, Chrysostomos STYLIOS et Peter P GROUMPOS, « Unsupervised learning techniques for fine-tuning fuzzy cognitive map causal links », in : *International Journal of Human-Computer Studies* 64.*8* (2006), p. 727-743.

[PK95] Kyung Sam PARK et Soung Hie KIM, « Fuzzy cognitive maps considering time relationships », in : *International Journal of Human-Computer Studies* 42.*2* (1995), p. 157-168.

[Pat13] Richard E PATTIS, *Ebnf : A notation to describe syntax*, 2013.

[Pea88] Judea PEARL, *Probabilistic Reasoning in Intelligent Systems : Networks of Plausible Inference*, San Francisco, CA, USA : Morgan Kaufmann Publishers Inc., 1988, ISBN : 0934613737.

[Ped+12] Vasco Calais PEDRO, Lucian Vlad LITA, Radu Stefan NICULESCU et R Bharat RAO, *System and method for creating and searching medical ontologies*, US Patent 8,229,881, juil. 2012.

[Pel16] Géraud Fokou PELAP, « Conception d'un famework pour la relaxation des requêtes SPARQL », thèse de doct., 2016.

[PLA13] A PEREGO, M LUTZ et P ARCHER, « ISA Programme Location Core Vocabulary », in : *EU ISA Programme Core Vocabularies Working Group (Location Task Force)* (2013).

[PAG09] Jorge PÉREZ, Marcelo ARENAS et Claudio GUTIERREZ, « Semantics and complexity of SPARQL », in : *ACM Transactions on Database Systems (TODS)* 34.*3* (2009), p. 1-45.

[Pet66] C. A. PETRI, *Communication with automata*, rapp. tech., 1966.

[PV11] Francois PICALAUSA et Stijn VANSUMMEREN, « What are real SPARQL queries like ? », in : *Proceedings of the International Workshop on Semantic Web Information Management*, 2011, p. 1-6.

[Pla19] Stefan PLANTIKOW, « Towards an International Standard for the GQL Graph Query Language », in : (2019).

[Poi06] Denis POIGNONEC, « Combining cognitive mapping and ontology to improve the understanding of the local stakeholders' perception of a coral reef ecosystem functioning in New Caledonia », Theses, Ecole Nationale Supérieure Agronomique de Rennes, déc. 2006, URL : `https://hal.ird.fr/tel-01171603`.

[Pa14] Mara POVEDA-VILLALÓN et AL, « A Pattern for Periodic Intervals », in : *Semantic Web Journal* (2014), p. 1-10.

[Pri+08] Magali PRIGENT, Guy FONTENELLE, Marie-Joëlle ROCHET et Verena M TRENKEL, « Using cognitive maps to investigate fishers' ecosystem objectives and knowledge », in : *Ocean & Coastal Management* 51.*6* (2008), p. 450-462.

[Pru08] Eric PRUD'HOMMEAUX, « SPARQL query language for RDF, W3C recommendation », in : *http ://www. w3. org/TR/rdf-sparql-query/* (2008).

[PS06] Eric PRUDHOMMEAUX et Andy SEABORNE, « Sparql query language for rdf. w3c working draft, 4 october 2006 », in : *SPARQL Query Language for RDF* (2006).

[RC89] David A RANDELL et Anthony G COHN, « Modelling Topological and Metrical Properties in Physical Processes. », in : *KR* 89 (1989), p. 357-368.

[RCC92] David A RANDELL, Zhan CUI et Anthony G COHN, « A spatial logic based on regions and connection. », in : *KR* 92 (1992), p. 165-176.

[Rec03] Alan RECTOR, « Description logics in medical informatics », in : *THE DESCRIPTION LOGIC HANDBOOK :Theory, implementation and applications*, 2003, chap. 13.

[Res+16] Oskar van REST, Sungpack HONG, Jinha KIM, Xuming MENG et Hassan CHAFI, « PGQL : a property graph query language », in : *Proceedings of the Fourth International Workshop on Graph Data Management Experiences and Systems*, 2016, p. 1-6.

[Rie76] Chuck RIEGER, « An organization of knowledge for problem solving and language comprehension », in : *Artificial Intelligence* 7.*2* (1976), p. 89-127, ISSN : 0004-3702, DOI : https://doi.org/10.1016/0004-3702(76)90001-1, URL : http://www.sciencedirect.com/science/article/pii/0004370276900011.

[RGL18a] Adrian ROBERT, David GENEST et Stéphane LOISEAU, « A Query Language for Cognitive Maps », in : *International Conference on Artificial Intelligence : Methodology, Systems, and Applications*, Springer, 2018, p. 218-227.

[RGL18b] Adrian ROBERT, David GENEST et Stéphane LOISEAU, « The Taxonomic Cognitive Map Query Language : A General Approach to Analyse Cognitive Maps », in : *2018 IEEE 30th International Conference on Tools with Artificial Intelligence (ICTAI)*, IEEE, 2018, p. 999-1006.

[RGL20a] Adrian ROBERT, David GENEST et Stéphane LOISEAU, « Cartes Cognitives Temporelles. », in : *EGC*, 2020, p. 309-316.

[RGL20b] Adrian ROBERT, David GENEST et Stéphane LOISEAU, « Temporal Cognitive Maps. », in : *ICAART (2)*, 2020, p. 58-68.

[Rob+19] Adrian ROBERT, David GENEST, Stéphane LOISEAU, Thomas RAIMBAULT et Brice TROUILLET, « Les cartes cognitives temporelles : modélisation et interrogation. », in : *EGC*, 2019, p. 369-370.

[Rod15] Marko A RODRIGUEZ, « The gremlin graph traversal machine and language (invited talk) », in : *Proceedings of the 15th Symposium on Database Programming Languages*, 2015, p. 1-10.

[Röh94] Ralf RÖHRIG, « A theory for qualitative spatial reasoning based on order relations », in : *Proceedings of the 12th national conference on AI (vol. 2)*, AAAI, 1994, p. 1418-1423.

[Sal+11] JM SALAS, A HARTH, B NORTON, LM VILCHES, AD LEÓN, J GOODWIN, C STADLER, S ANAND et D HARRIES, « NeoGeo Vocabulary : Defining a shared RDF representation for GeoData », in : *Public draft, NeoGeo, May* (2011).

[SL99] Richard SATUR et Zhi-Qiang LIU, « A contextual fuzzy cognitive map framework for geographic information systems », in : *IEEE transactions on fuzzy systems* 7.*5* (1999), p. 481-494.

[SS91] Manfred SCHMIDT-SCHAUSS et Gert SMOLKA, « Attributive concept descriptions with complements », in : *Artificial intelligence* 48.*1* (1991), p. 1-26.

[Sco82] Dana S SCOTT, « Domains for denotational semantics », in : *International Colloquium on Automata, Languages, and Programming*, Springer, 1982, p. 577-610.

[SV95] Moninder SINGH et Marco VALTORTA, « Construction of Bayesian network structures from data : a brief survey and an efficient algorithm », in : *International journal of approximate reasoning* 12.*2* (1995), p. 111-131.

[Sir+07] Evren SIRIN, Bijan PARSIA, Bernardo Cuenca GRAU, Aditya KALYANPUR et Yarden KATZ, « Pellet : A practical owl-dl reasoner », in : *Journal of Web Semantics* 5.*2* (2007), p. 51-53.

[Sow76] John F SOWA, « Conceptual graphs for a data base interface », in : *IBM Journal of Research and Development* 20.*4* (1976), p. 336-357.

[Sow87] John F SOWA, « Semantic networks », in : (1987).

[SRA90] Sampath SRINIVAS, Stuart RUSSELL et Alice AGOGINO, « Automated construction of sparse Bayesian networks from unstructured probabilistic models and domain information », in : *Machine Intelligence and Pattern Recognition*, t. 10, Elsevier, 1990, p. 295-308.

[SKP05] Wojciech STACH, LA KURGAN et Witold PEDRYCZ, « A survey of fuzzy cognitive map learning methods », in : *Issues in soft computing : theory and applications* (2005), p. 71-84.

[SGB00] Robert STEVENS, Carole A GOBLE et Sean BECHHOFER, « Ontology-based knowledge representation for bioinformatics », in : *Briefings in bioinformatics* 1.*4* (2000), p. 398-414.

[Ten76] Robert D. TENNENT, « The denotational semantics of programming languages », in : *Communications of the ACM* 19.*8* (1976), p. 437-453.

[Tol48] Edward C TOLMAN, « Cognitive maps in rats and men. », in : *Psychological review* 55.*4* (1948), p. 189.

[TM97] Athanasios K TSADIRAS et Konstantinos G MARGARITIS, « Cognitive mapping and certainty neuron fuzzy cognitive maps », in : *Information Sciences* 101.*1-2* (1997), p. 109-130.

[UK95] Michael USCHOLD et Martin KING, *Towards a methodology for building ontologies*, Citeseer, 1995.

[Van+19] Linda VAN DEN BRINK et al., « Best practices for publishing, retrieving, and using spatial data on the web », in : *Semantic Web* 10.*1* (2019), p. 95-114.

[Ver14] Annemieke R VERBOON, « The medieval tree of Porphyry : An organic structure of logic », in : *The tree : Symbol, allegory, and mnemonic device in medieval art and thought*, 2014, p. 95-116.

[Vie04] AC VarziandL VIEU, « Searching for a time ontology for semantic web applications », in : *Formal Ontology in Information Systems : Proceedings of the Third International Conference (FOIS-2004)*, IOS Press, 2004, p. 331.

[Wal47] Arthur Geoffrey WALKER, « Durées et instants », in : *Revue Scientifique* 85 (1947), p. 131-134.

[Woo75] William A WOODS, « What's in a link : Foundations for semantic networks », in : *Representation and understanding*, Elsevier, 1975, p. 35-82.

[Wro+03] Chris WROE, Robert STEVENS, Carole GOBLE, Angus ROBERTS et Mark GREENWOOD, « A suite of DAML+ OIL ontologies to describe bioinformatics web services and data », in : *International Journal of Cooperative Information Systems* 12.*02* (2003), p. 197-224.

[YJ16] Byung Sung YOON et Antonie J JETTER, « Comparative analysis for fuzzy cognitive mapping », in : *2016 Portland International Conference on Management of Engineering and Technology (PICMET)*, IEEE, 2016, p. 1897-1908.

[Zad88] Lotfi A ZADEH, « Fuzzy logic », in : *Computer* 21.*4* (1988), p. 83-93.

[Zha91] Jiajie ZHANG, « The interaction of internal and external representations in a problem solving task », in : *Proceedings of the thirteenth annual conference of cognitive science society*, Erlbaum Hillsdale, NJ, 1991, p. 954-958.

[Zho+08] Haoming ZHONG, Chunyan MIAO, Zhiqi SHEN et Yuhong FENG, « Temporal fuzzy cognitive maps », in : *2008 IEEE International Conference on Fuzzy Systems (IEEE World Congress on Computational Intelligence)*, IEEE, 2008, p. 1831-1840.

[ZF02] Qing ZHOU et Richard FIKES, « A reusable time ontology », in : *Proceeding of the AAAI Workshop on Ontologies for the Semantic Web*, 2002.

UNIVERSITE BRETAGNE LOIRE / MATHSTIC

Titre : Représentation et interrogation de connaissances dans des cartes cognitives.

Mot clés : Représentation des connaissances, Cartes cognitives, Représentation du temps et de l'espace, Langages de requête

Resumé : Le modèle des cartes cognitives est un modèle de représentation des connaissances qui vise à représenter des systèmes d'influences complexes. Il est simple et visuel mais assez formel pour être manipulé de manière informatique. L'enjeu de cette thèse réside dans l'idée d'améliorer les capacités informatiques du modèle sans trop le complexifier pour ne pas le dénaturer. Cette thèse se base sur un cas réél d'utilisation du modèle dans le milieu de l'halieutique ; elle propose deux contributions. La première consiste à étendre le modèle en y associant une ontologie de manière à pouvoir caractériser les concepts utilisés sur leurs aspects spatiaux et temporels. La seconde contribution consiste à interroger les connaissances du modèle au travers d'un langage de requête. Ce langage se base sur des relations atomiques, appelées primitives, pour accèder à des connaissances spécifiques qui peuvent être exprimées de manière déclarative. Cette thèse propose également un prototype regroupant l'implémentation des contributions.

Title : Representation and extraction of knowledge in cognitive maps.

Keywords : Knowledge representation, Cognitive map, Spatial and temporal representation, Query languages

Abstract : The cognitive map model is a visual knowledge representation model which aims to model complex influence systems. The whole point of this thesis is to enhance the computational capababilities of the model while keeping it simple and easy to use. This thesis is based on a real use case and proposes 2 contributions. The first one extends the expressivity of the model by linking it to an ontology that is used to describe the concepts of the model on their spatial and temporal aspects. The second contribution is a query language for cognitive maps, called CMQL. This language is based on atomic relations called 'primitives' to access the specified knowledge. This thesis also proposes a prototype that implements the diferents contributions.