THÈSE DE DOCTORAT DE L'UNIVERSITÉ SORBONNE UNIVERSITÉ

Spécialité

Informatique

École doctorale Informatique, Télécommunications et Électronique (Paris)

Présentée par

Adam FACI

Pour obtenir le grade de

DOCTEUR de SORBONNE UNIVERSITÉ

Sujet de la thèse :

Représentation, simulation et exploitation de connaissances dans le formalisme des graphes conceptuels

Table des matières

Remerciements

Je remercie tout d'abord et avant tout ma directrice de thèse Marie-Jeanne Lesot et mon encadrante Claire Laudy. Elles m'ont grandement aidé, soutenu, guidé, formé, supporté et critiqué. Nos rapports ont mûri en une collaboration entre collègues de métier, pouvant déterminer conjointement des directions de recherche et affiner une proposition scientifique. Au delà de ces aspects, je les apprécie surtout sur le plan humain, elle ont été un grand soutien moral et je valorise beaucoup leur éthique et spontanéité.

Je remercie ensuite les collègues de l'équipe LFI et de Thales, et en particulier Christophe Marsala pour ses relectures et ses commentaires, et Nicolas Museux pour ses questions pertinentes et sa persistance.

Les copains et copines thésard•e•s ont été mes plus fines équipes de rigolade et de bonne mangeaille, avec d'une part Douae, Quentin, Paul, Erwann et Roman, et d'autre part Marcin, Ivo, Arthur, Thibault, Jérémie, Clara, Adulam, Garance et Yann.

Surtout je remercie Marcin et Arthur pour leur bonne camaraderie et leur astuces en début de thèse ; et Thibault et Jérémie pour leur compagnonnage de chaque instant.

Enfin je remercie ma famille pour leur soutien moral et leur compréhension, et surtout "Papa à", Lili, Vivi, Mayou, Hanna, Sohane et Dina.

Et merci à Anouk le pilier de ces 3 années, et Émile pour son titre alternatif fort original mais malheureusement trop peu pertinent : « L'Apocalypse des Robots ».

Introduction générale

Contexte

Dans de très nombreux cas applicatifs, des experts disposent de grands volumes de connaissances et de données collectées. Une des questions cruciales qui se posent dans ce cadre est l'exploitation de ces connaissances pour permettre la construction de nouvelles connaissances.

Cette question se décline différemment selon le cadre applicatif dans lequel elle s'inscrit. Un des choix essentiels est celui du formalisme de représentation des connaissances qui doit optimiser leur expressivité et leur interprétabilité, et doit fournir des outils d'exploitation riches et efficaces.

Les travaux présentés dans cette thèse se placent dans le cadre des graphes conceptuels (Chein & Mugnier, 2008), qui constituent un formalisme structuré de représentation des connaissances. Il distingue différentes sortes de connaissances : les connaissances ontologiques, énoncées sous la forme d'un vocabulaire (que l'on peut considérer comme un ensemble de connaissances terminologiques) et les connaissances factuelles, exprimées dans le langage défini par ce vocabulaire. Les graphes conceptuels proposent en outre une forme graphique visuelle qui accroît potentiellement le degré d'interprétabilité des connaissances représentées. Ils offrent aussi des outils de manipulation efficaces, basés sur la théorie des graphes, qui permettent une représentation visuelle de leur exploitation.

D'un point de vue applicatif, il existe une multitude d'utilisations des graphes conceptuels parmi lesquelles on peut citer les procédés cliniques (Kamsu-Foguem et al., 2014), la sécurité (Fu et al., 2017), la finance (Kamaruddin et al., 2008), le clustering (Ganascia & Velcin, 2004), la prise de décision (Tremblay et al., 2017). la comparaison d'une musique expérimentale et d'une technique de jardinage traditionnelle japonaise (Fowler, 2019), l'épistémologie (Laudy et al., 2009), la représentation de traçabilité (Laudy & de Naurois, 2021), de flux RSS (Gkiokas & Cristea, 2014), de médecine traditionnelle africaine (Kamsu-Foguem et al., 2013), de logiciels (Vlasenko et al., 2019) ou de programmes télévisés (Laudy et al., 2007).

Ainsi, Kamsu-Foguem et al.(2014) représentent des protocoles cliniques par des graphes conceptuels, permettant aux utilisateurs de comprendre intuitivement les étapes des raisonnements qui sont appliqués. Fowler(2019) modélisent puis comparent une pièce de musique expérimentale et une technique de jardinage traditionnelle japonaise sous forme de graphes conceptuels. Les raisonnements du formalisme permettent d'inférer de nouvelles connaissances et la relative facilité de traduction depuis des connaissances exprimées en langage naturel y est valorisée.

Travaux de thèse

Inscrits dans ce formalisme des graphes conceptuels, nos travaux de thèse, menés dans le cadre d'un financement Cifre en collaboration avec THALES TRT et l'équipe LFI du laboratoire LIP6, portent sur trois problématiques d'intelligence artificielle symbolique : la représentation, la simulation et l'exploitation de connaissances.

En premier lieu, nous nous intéressons au problème théorique concernant la capacité à représenter des connaissances expertes dans ce formalisme expressif. Nous considérons la question de la représentation des connaissances, et en particulier des connaissances imprécises, de manière à accroître expressivité et interprétabilité du formalisme, notamment par la définition de poids ou l'intégration d'ensembles flous, qui quantifient l'imprécision. Dans ce but, nous proposons une étude comparative de l'état de l'art sur les extensions pondérées des graphes conceptuels, basées sur la logique floue. Ces travaux ont fait l'objet d'une publication dans SSCI-FOCI 2021 (Faci et al., 2021c).

Ensuite, nous considérons le problème algorithmique et pratique lié au besoin de disposer de *benchmarks* pour la validation d'algorithmes exploitant les connaissances. Nous explorons deux types d'approches pour la construction de bases de graphes conceptuels, basés sur la traduction et la génération à partir de modèles de connaissance suivant différentes contraintes. Une publication dans la conférence IFSA-EUSFLAT 2021 (Faci et al., 2021a) est issue de cette étude.

Enfin, nous nous intéressons au problème théorique et algorithmique induit par l'accumulation exponentielle de connaissances qui deviennent difficilement interprétables, et qui nécessitent le développement d'outils automatiques pour les filtrer et les agréger. Nous abordons la question de l'exploitation de ces connaissances, sous l'angle de l'extraction de motifs fréquents d'intérêt. Nous proposons un algorithme d'extraction de motifs fréquents dans des graphes conceptuels, minimisant la redondance avec les connaissances ontologiques, et employant leurs particularités pour accroître son efficacité. Ce travail a donné lieu à un article à la conférence ICAISC 2021 (Faci et al., 2021b). Enfin, nous validons ce même algorithme sur une problématique industrielle en répondant à un

besoin métier de caractérisation de trajectoires pour la sécurité maritime, mettant ainsi en application nos travaux théoriques sur un cas pratique réel.

Structure du manuscrit

Le manuscrit reprend la structure des contributions présentées ci-dessus.

Tout d'abord en partie I, le cadre théorique dans lequel nous nous plaçons est étudié, par la description du formalisme des graphes conceptuels et ses ancêtres dans le chapitre 1. Une étude comparative des modèles de graphes conceptuels flous est ensuite présentée dans le chapitre 2, page 37. Cette étude ayant été effectuée en fin de thèse suite aux questions ouvertes par les parties suivantes, ces dernières se limitent aux cadre non-flou des graphes conceptuels. Les modèles de graphes conceptuels flous sont néanmoins présentés ici car traitent de représentation de connaissances.

La partie II est consacrée au problème de la simulation de connaissances : dans le chapitre 3, page 77, nous abordons la question de la traduction de connaissances depuis un langage du Web sémantique, *RDF/RDF(S)*, vers le formalisme des graphes conceptuels. Nous étudions ensuite, dans le chapitre 4, page 97, le cas de la simulation de connaissances par génération à partir de contraintes ontologiques.

Enfin en partie III, nous nous intéressons à l'exploitation de ces connaissances avec la proposition d'un algorithme d'extraction de motifs (chapitre 5, page 119), validé sur les données générées par les approches de la partie II puis par un cas industriel réel (chapitre 6, page 139).

Nous terminons par une conclusion suivie d'un ensemble de perspectives.

Première partie

Représentation de la connaissance

La représentation de la connaissance est une question centrale en informatique : elle vise à permettre l'exploitation de connaissances par un système d'information. Il est nécessaire d'assurer un équilibre entre l'expressivité du formalisme choisi, et sa complexité pour l'exploitation et l'interprétation des connaissances ainsi représentées.

La recherche d'expressivité d'un formalisme correspond à l'ajout d'éléments permettant de représenter différentes sortes d'informations, à plusieurs niveaux et selon des sémantiques variées, ce qui augmente le temps d'interprétation des connaissances ainsi modélisées.

La recherche de simplicité d'un formalisme correspond à une modélisation minimisant le temps d'exploitation d'interprétation d'un maximum d'information. Cela peut être discuté autant du point de vue d'un système informatique que par celui d'humains, experts ou non.

Cette partie discute d'une famille de formalismes représentant les connaissances de manière structurée, les graphes conceptuels, ainsi que leur extensions permettant la représentation de connaissances imparfaites. Ils sont au cœur de nos thématiques de recherche sur la simulation et l'exploitation de connaissances des parties II et III. Nous détaillons ici le contexte dans lequel ils ont été définis, leurs spécificités par rapport aux autres formalismes et les possibilités de représentation qu'ils offrent.

Le chapitre 1 effectue un bref rappel de la logique floue sur laquelle les extensions des graphes conceptuels pour la représentation de connaissances imparfaites sont construites. Ensuite, différents formalismes de représentation de la connaissance structurée sont présentés. Une partie de ces formalismes ont précédé les graphes conceptuels : ces derniers ont alors tenté de répondre à des lacunes des premiers. D'autres sont contemporains voire postérieurs au formalisme des graphes conceptuels mais cherchent à garantir des propriétés différentes. Enfin, la famille de formalismes des graphes conceptuels est présentée, avec une emphase sur le formalisme des graphes conceptuels basiques de Chein et Mugnier, 2008.

Le chapitre 2, page 37 recense les différentes propositions, basées sur des méthodes issues de la logique floue, concernant les extensions pondérées des graphes conceptuels visant à représenter des connaissances imparfaites. Nous y réalisons une étude comparative des modèles de graphes conceptuels flous qui en résultent. Cette étude consiste en un état des lieux des graphes conceptuels flous au prisme de plusieurs axes : la partie du formalisme concernée par l'extension floue, la contrainte de représentation relâchée ainsi que le type de pondération utilisé.

Chapitre 1

Connaissances imparfaites et connaissances structurées

Ce chapitre présente le cadre général dans lequel s'inscrivent nos travaux, en rappelant les notations et définitions des formalismes considérés pour la représentation des connaissances : nous y considérons successivement les cas où celles-ci sont imparfaites car imprécises puis les cas où elles sont munies d'une structure qui facilite leur exploitation et interprétation.

Ces différents formalismes proposent de répondre au problème, crucial en intelligence artificielle symbolique, de la représentation des connaissances. Elles proposent différents compromis entre expressivité et qualité des outils de manipulation associés, cette dernière couvrant notamment l'efficacité, par exemple en terme de coût de calcul, ou l'existence de garantie sur la cohérence et correction des inférences réalisées.

Ce chapitre rappelle d'abord, dans la section 1.1, le cadre de la logique floue, formalisme qui étend la logique des prédicats du premier ordre classique pour augmenter son expressivité. Il décrit ensuite, section 1.2, page 21, le cadre des graphes conceptuels qui enrichit la représentation en intégrant des informations de structure et en combinant le formalisme de la logique des prédicats du premier ordre et la théorie des graphes.

1.1 Formalisme logique pour la représentation de connaissances imprécises : la logique floue

La logique classique constitue un formalisme de représentation des connaissances muni de règles de raisonnement sur ces connaissances, qui permettent d'en inférer de nouvelles. La logique des prédicats du premier ordre, qui n'autorise que deux valeurs de vérité, vrai ou faux représentés par $\{0, 1\}$, a été enrichie pour représenter des connais-

sances imparfaites, par exemple incertaines ou imprécises : les premières sont associées à des valeurs de vérité telles que *peut-être vrai* ou *certainement faux* ; les secondes à *partiellement vrai* ou *presque totalement faux*. Elles se basent sur des pondérations numériques qui étendent les valeurs de vérité de $\{0,\ 1\}$ à $[0,\ 1]$, selon une sémantique possibiliste ou floue (Dubois & Prade, 1993, 2012).

Cette section présente le second cadre, la logique floue, introduit par Zadeh, 1965, puis étendu ensuite par des modèles recensés par exemple par Dubois et Prade, 2015 ou Bouchon-Meunier, 2007. Elle a été proposée pour représenter des connaissances et des raisonnements de manière intuitive, offrant plus de flexibilité que les raisonnements basés sur les outils mathématiques de la logique des prédicats du premier ordre, et permet aussi d'offrir une interprétabilité accrue des manipulations de la connaissance. Sa sémantique repose sur la théorie des sous-ensembles flous, brièvement rappelée dans la section 1.1.1 ; ses caractéristiques principales sont ensuite présentées dans la section 1.1.2.

1.1.1 Théorie des sous-ensembles flous

La théorie des sous-ensemble flous est une généralisation de la théorie des ensembles classiques permettant la représentation de connaissances imparfaites. Cette généralisation s'effectue par l'utilisation d'un degré d'appartenance à un ensemble au lieu de se limiter au cas binaire classique de la relation $\in$.

Elle permet en particulier la représentation de connaissances imprécises ou vagues, de situations intermédiaires ou de catégories mal définies. Par exemple, l'appartenance d'un individu aux catégories *jeune* ou *riche* n'est pas déterminée de manière binaire, et la logique floue ici permet de représenter une appartenance partielle ou graduelle selon respectivement l'âge et les revenus de l'individu.

Formellement, on considère un univers de discours U. Un sous-ensemble classique C de U est défini par sa fonction caractéristique $\chi_C : U \longrightarrow \{0,1\}$. Un sous-ensemble flous F de U quant a lui est défini par sa fonction d'appartenance $\mu_F : U \longrightarrow [0,1]$. Cette fonction d'appartenance admet toutes les valeurs réelles comprises entre 0 et 1, contrairement à la fonction caractéristique classique.

Une variable linguistique est un triplet (V, U_V, K_V) tel que V est une variable, U_V l'univers sur lequel V est définie, et K_V un ensemble de modalités linguistiques. Les modalités sont représentées par des sous-ensembles flous définis par leurs fonctions d'appartenance sur U_V et par des étiquettes linguistiques.

La figure 1.1 représente le cas où $V = \mathit{ProportionÉtudiant(e)s}$, $U_V = [0,1]$ et $K_V = \{\mathit{Peu}, \mathit{Beaucoup}\}$. Selon la proportion d'étudiants et d'étudiantes d'un groupe donné, il est ainsi caractérisé comme représentant peu à beaucoup d'étudiants.

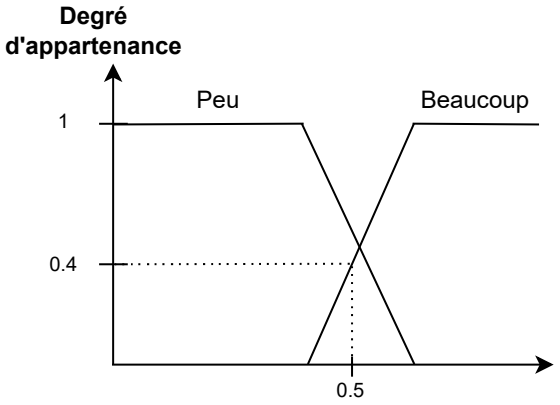


FIGURE 1.1 – Sous-ensemble flou représentant les quantificateurs flous *Peu* et *Beaucoup* définis sur l'univers de proportions $[0,1]$

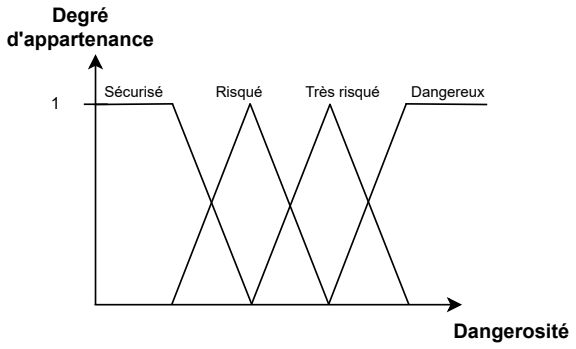


FIGURE 1.2 – Modalités floues de la variable linguistique *Dangerosité*

La figure 1.2 représente le cas où $V = Dangerosité$ et

$$K_V = \{Sécurisé, Risqué, TrèsRisqué, Dangereux\}$$

dans le contexte d'évaluation du danger d'une attaque informatique. La valeur de dangerosité n'est pas explicite mais elle pourrait être calculée selon une combinaison de facteurs tels que la compétence de l'attaquant, la vulnérabilité de la cible et son importance.

1.1.2 Logique floue

La logique floue (ZADEH, 1965) est une extension de la logique des prédicats du premier ordre, où la sémantique d'un prédicat n'est pas définie comme un sous-ensemble classique de son univers associé, mais comme un sous-ensemble flou. Ainsi, alors qu'un prédicat, noté p, est associé à une fonction $X_p : U \longrightarrow \{0, 1\}$ dans le cadre classique, il est associé à une fonction $\mu_p : U \longrightarrow [0, 1]$ dans le cadre flou.

Par exemple, en reprenant la figure 1.2, on y dispose d'un ensemble de prédicats :

$$\{Sécurisé, Risqué, TrèsRisqué, Dangereux\}$$

de fonctions d'appartenance respectives :

$$\{\mu_{Sécurisé}, \mu_{Risqué}, \mu_{TrèsRisqué}, \mu_{Dangereux}\}$$

sur l'univers U des niveaux de *Dangerosité*.

Par ailleurs, en considérant le cas d'un univers catégoriel :

$$U = \{Histoire, Géographie, Mathématiques, Informatique\}$$

des cours dispensés dans une université, on peut définir la fonction d'appartenance de *CoursPopulaire* par :

$$\begin{array}{rccl} \mu_{CoursPopulaire} : & U & \longrightarrow & [0,1] \\ & \text{Histoire} & \mapsto & 1 \\ & \text{Géographie} & \mapsto & 0.4 \\ & \text{Mathématiques} & \mapsto & 0.6 \\ & \text{Informatique} & \mapsto & 0.7 \end{array}$$

Le calcul de vérité d'une formule logique est étendu pour correspondre à la nouvelle sémantique. Notons x une variable à valeurs dans l'univers de discours U. La valeur de vérité de la formule atomique $p(x)$ est définie comme le degré d'appartenance de la valeur $v(x)$ donnée à x au sous-ensemble flou qui définit p :

$$[p(x)] = \mu_p(v(x))$$

où $\mu_p(v(x))$ renvoie le degré d'appartenance de $v(x)$ à p.

Ensuite, le calcul pour chaque connecteur logique est adapté, avec par exemple pour la conjonction de deux formules logiques F_1 et F_2 :

$$[F_1 \wedge F_2] = \top([F_1], [F_2])$$

où $\top$ est la t-norme, dont un exemple d'instanciation est la fonction *min* (Zadeh, 1965). Il en est de même pour les opérateurs de disjonction, implication, négation, ainsi que les quantificateurs $\forall$ et $\exists$. Leur extension n'est pas détaillée ici, Kaufmann et Gupta, 1986 en offrent une formalisation.

Enfin, les outils de raisonnement sont également étendus dans le cadre flou. Par exemple, le Modus Ponens dans le cadre non flou établit que< de $A \implies B$ et A, on

peut inférer B. Dans le cadre flou, le Modus Ponens Généralisé (Zadeh, 1975) est défini, et il établit que de $A \implies B$ et A', qui peut différer de A, on peut inférer B' en explicitant sa fonction d'appartenance en fonction de celles de A, B et A'.

Cette extension permet d'accroître l'expressivité des connaissances représentées. Leur exploitation en devient plus difficile, néanmoins leur interprétabilité en devient plus aisée (Jin, 2000 ; Lesot et al., 2016).

1.2 Formalisme graphique pour la représentation de connaissances structurées : les graphes conceptuels

Un autre compromis entre expressivité et complexité d'un formalisme est de reprendre la logique des prédicats du premier ordre et de la munir d'outils de manipulation plus puissants, tels que la théorie des graphes. Ainsi son expressivité reste identique mais l'efficacité des algorithmes et l'interprétabilité des connaissances en sont augmentées.

Les graphes conceptuels sont une famille de formalismes de représentation structurée de la connaissance qui héritent de plusieurs formalismes antérieurs. Ils répondent au problème de complexité par un accès aux puissants algorithmes de la théorie des graphes et l'interprétabilité de leur représentation graphique.

Ils ont été proposés en premier par Sowa, 1983 et sont basés sur logique des prédicats du premier ordre (Barwise, 1977) et certaines de leurs extensions sur la logique floue. Ces dernières permettent de représenter des connaissances de manière formelle et de raisonner avec ces connaissances, comme vu dans la section précédente. Cependant, les mécanismes de raisonnement posent des problèmes d'efficacité, et il est par exemple difficile de passer à l'échelle lorsque la quantité d'opérations à effectuer croît.

Les graphes conceptuels remédient à cet inconvénient en permettant une organisation des connaissances de manière structurée, à l'interface entre la logique des prédicats et la théorie des graphes. Cette structuration assure à la fois une manipulation plus efficace, grâce à la multitude d'algorithmes fonctionnant sur des graphes, et une représentation graphique explicite, ce qui résulte en une meilleure interprétabilité des connaissances ainsi représentées.

Les différents principes fondant les graphes conceptuels sont abordés successivement par une description brève des formalismes dont ils héritent en section 1.2.1. Puis nous présentons le cas des graphes conceptuels, qui offrent en particulier l'avantage de séparer différentes sortes de connaissances : d'une part des connaissances ontologiques et d'autre part des connaissances factuelles. Les connaissances ontologiques (section 1.2.2, page 24) consistent en la représentation d'informations terminologiques qui définissent ce qui peut être représenté, et sont définies dans le cas des graphes concep-

tuels par un vocabulaire. Les connaissances factuelles (section 1.2.3, page 28) quant à elles utilisent les éléments définis dans le vocabulaire pour représenter des informations. Le chapitre 2, page 37 discute d'extensions des graphes conceptuels où la logique sous-jacente est la logique floue.

La section 1.2.4, page 30 présente l'une des extensions du cadre de base des graphes conceptuels qui est utilisée dans la suite de la thèse : les règles-λ. Elles constituent une troisième sorte de connaissances, dites implicites, parfois comprises comme un enrichissement des connaissances ontologiques, représentant des règles d'inférence, des transformations possibles, ou des contraintes (Chein & Mugnier, 2009).

Enfin, la section 1.2.5, page 33 présente une des tâches à la base de la plupart des mécanismes de raisonnement dans les graphes conceptuels, la recherche d'homomorphisme. Elle fonde le socle théorique pour la simulation de connaissances à partir de génération du chapitre 4, page 97 et de l'extraction de motifs fréquents dans une base de graphes conceptuels du chapitre 5, page 119. En effet, les opérations de fusion de graphe et de calcul de support d'un motif qui y sont utilisées correspondent à une recherche d'homomorphisme entre d'une part les parties de graphes fusionnées, et d'autre part entre le motif et les sous-graphes qui le supportent.

1.2.1 Une dynastie de formalismes structurés

Les graphes conceptuels sont basés sur les réseaux sémantiques (Sowa, 1987 ; Lehmann, 1992) ainsi que les graphes existentiels de C. S. Peirce (Sowa, 2011), qui sont les premiers formalismes de représentations structurées. Il est à noter que des apparitions de représentation de connaissances sous forme de diagrammes existent jusque dans la Grèce antique (Verboon, 2014), mais n'ont à notre connaissance pas fait l'objet d'un travail de formalisation aussi poussé.

Réseaux sémantiques

Les réseaux sémantiques (Sowa, 1987 ; Lehmann, 1992) constituent une des premières formalisations de la représentation de la connaissance sous forme structurée. Ils se fondent sur une version simplifiée de la logique des prédicats du premier ordre en se munissant d'une représentation sous forme de graphes étiquetés. Ils considèrent un ensemble d'objets, des concepts au sein d'une ontologie, interconnectés par des arcs selon trois sémantiques : les relations *isA*, *hasA* et *isAKindOf* respectivement correspondant à l'instanciation, la composition et la subsomption. D'autres types de relations peuvent également être définis entre ces concepts pour représenter des connaissances plus spécifiques, mais aucune distinction n'est effectuée entre connaissance ontologique et factuelle.

Graphes existentiels

Les graphes existentiels (Sowa, 2011) sont un formalisme de représentation diagrammatique de connaissance et de raisonnement. Ils étendent les réseaux sémantiques en un système de représentation et de manipulation des connaissances plus riche et plus intuitif. En particulier, ce formalisme correspond à un modèle logique dont la syntaxe et la sémantique sont des éléments graphiques tels que la page blanche, le cercle ou des mots et caractères : une page blanche correspond à l'absence de connaissance, le cercle à la négation de tout ce qu'il entoure, et les mots et caractères sont des identifiants. Un des objectifs est de permettre une plus grande clarté et une plus grande interprétabilité des connaissances et raisonnements logiques. Ceci est notamment permis par la représentation diagrammatique, jugée plus facilement interprétable que des symboles mathématiques, ainsi qu'un mécanisme de raisonnement compris comme une succession d'opérations simples sur les diagrammes.

Logiques de description

Les logiques de description (Baader et al., 2017) sont une autre famille de formalismes basés sur la logique des prédicats du premier ordre. Elles proposent un compromis entre expressivité et complexité des raisonnements reposant sur la restriction des prédicats, dont l'arité est limitée à deux, et des utilisations des quantificateurs universel et existentiel.

Les premières logiques de description ont reposé sur une forme graphique, mais les plus récentes ont perdu cette formulation d'après Chein et Mugnier, 2008, contrairement aux graphes conceptuels.

Le Web sémantique et ses différents langages

Les langages du Web sémantique constituent un cas particulier de représentation de la connaissance et proposent entre autres d'établir un langage unifié d'annotation de documents et de ressources Web.

Le langage *RDF/XML* (Manola et al., 2004), pour *Resource Description Framework* sous forme de documents *XML*, en est un des principaux représentants. Les connaissances sont représentées sous la forme de bases de triplets (s, p, o), où s est le sujet du triplet, p le prédicat et o l'objet du triplet. Un tel triplet correspond à la formule logique atomique $p(s, o)$. Les composants s, p et o sont des URI, c'est-à-dire des identifiants uniques qui sont usuellement un lien URL menant à une page décrivant la ressource représentée, ou un identifiant anonyme, noté *blank*, dont l'interprétation peut varier selon la base de connaissances qui l'utilise.

Ce langage représente essentiellement des connaissances assertionnelles, c'est-à-dire des connaissances factuelles. Il a ensuite été étendu par la proposition de nouveaux langages permettant un certain nombre de déclarations terminologiques, comme le langage *RDF Schema* (Brickley et al., 2014), noté *RDF(S)*, et le langage *OWL* (McGuinness & Van Harmelen, 2004) (Web Ontology Language). Le langage *RDF(S)* permet la représentation d'une ontologie simple par l'ajout de classes, de leur hiérarchie et de signatures de relation ; le langage *OWL* ouvre plus de possibilités pour la construction d'une ontologie plus complète.

Le langage *RDF/XML*, adjoint du langage *RDF(S)* et/ou *OWL*, est similaire aux réseaux sémantiques, mais propose en plus le développement de conventions communes de représentation, une plus grande richesse des relations entre ressources et un contexte d'utilisation différent : alors que les réseaux sémantiques ont été pensés à l'origine pour représenter des taxonomies, le langage *RDF/XML* et ses extensions visent à représenter toute sorte de connaissance de manière unifiée.

Toutefois, comme discuté dans le chapitre 3, page 77, de manière similaire aux réseaux sémantiques, la différence entre connaissance ontologique et connaissance factuelle peut être ambiguë. En effet, l'utilisation très peu contrainte de ces langages peut conduire à des problèmes de raisonnements : les graphes conceptuels, au contraire, ne permettent pas ces ambiguïtés.

1.2.2 Connaissances ontologiques : le vocabulaire

Cette section et la suivante s'appuient sur la présentation des graphes conceptuels proposée par Chein et Mugnier, 2008.

Les graphes conceptuels (Chein & Mugnier, 2008) héritent des formalismes décrits dans la section précédente, en ce qu'ils proposent également de combiner une restriction de formules de la logique des prédicats du premier ordre, et de par leur représentation des connaissances sous forme de graphes étiquetés. Cette section et les suivantes en présentent les principales composantes, en distinguant les différentes sortes de connaissances qu'ils permettent de représenter, en particulier les connaissances ontologiques et factuelles respectivement. Nous détaillons dans cette section les principes et les objectifs de cette distinction puis rappelons la définition formelle du *vocabulaire* qui représente les connaissances ontologiques. Les connaissances factuelles sont présentées dans la section 1.2.3, page 28.

Principe et objectif

Les connaissances ontologiques expriment des connaissances générales sur l'univers, là où les connaissances factuelles décrivent des faits à propos d'entités spécifiques, com-

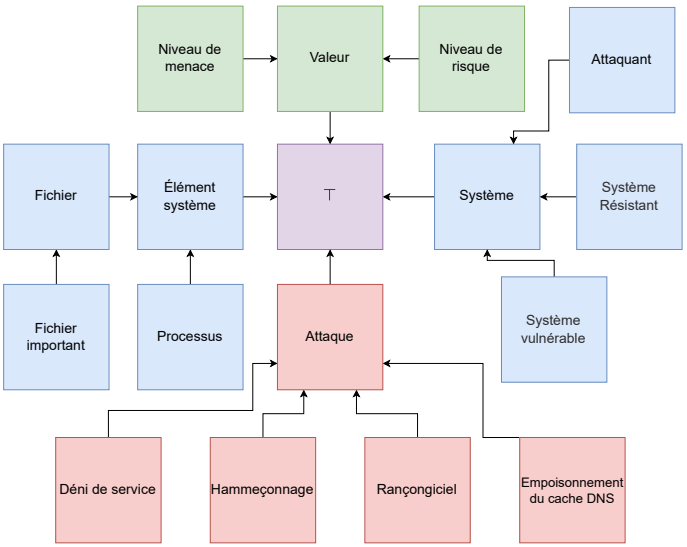


FIGURE 1.3 – Hiérarchie des types de concepts des attaques informatiques

me l'existence de ces entités, les catégories auxquelles elles appartiennent ou leurs relations : les connaissances ontologiques décrivent des méta-connaissances sur les informations représentées dans les connaissances factuelles.

On peut également formuler les connaissances ontologiques comme une spécification terminologique qui détaille les termes utilisables et leurs propriétés générales.

Plus généralement, les connaissances ontologiques consistent en la définition d'un vocabulaire qu'on peut utiliser pour décrire : les concepts dont les objets sont des instances, les propriétés que les objets possèdent et les relations entre les objets. Par ailleurs, les connaissances ontologiques comprennent également un ensemble de méta-objets qui réunit caractéristiques générales des concepts, propriétés ou relations, avec par exemple la définition d'une relation d'ordre entre ces méta-objets, ou la définition de connaissances partagées par leurs instances.

Par exemple, si l'on veut représenter des attaques informatiques, les connaissances ontologiques peuvent être définies comme les différents concepts représentant des attaques tels que *rançongiciel*, *hammeçonnage* ou *déni de service* par exemple, mais également d'autres concepts avec lesquels ces attaques sont en relation comme *fichier* ou *processus*. Ils peuvent être organisés en une hiérarchie telle que représentée sur la figure 1.3 et formalisée ci-dessous. Les relations entre les instances de ces concepts peuvent être des relations binaires comme une *action* d'un *processus* sur un *fichier*, qui peut être spécialisé en *supprime*, *copie* et *déplace* tel que représenté dans la hiérarchie figure 1.4, page 26. Cet exemple est complété avec des relations ternaires plus loin dans le chapitre.

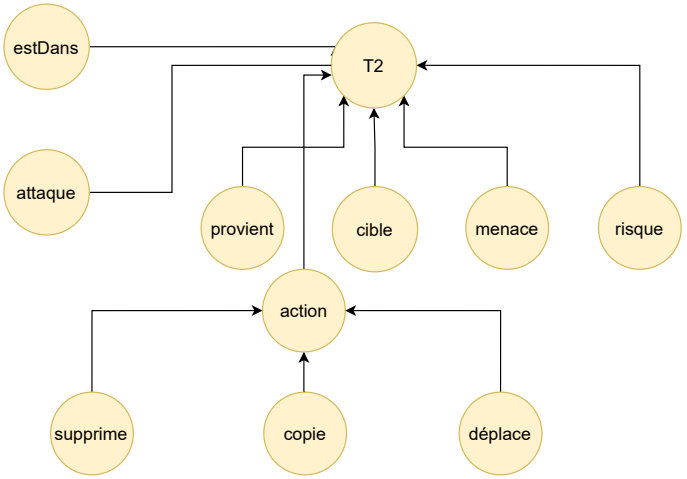


FIGURE 1.4 – Hiérarchie des types de relations des attaques informatiques

Exemples illustratifs

Nous considérons de plus l'exemple moins complexe dont l'objectif est de représenter la connaissance factuelle décrite linguistiquement par la phrase : *Nouka est une étudiante qui assiste au cours* 1H001 *qui est un cours d'histoire*. Les connaissances ontologiques liées à cet exemple sont les notions d'*humain*, d'*étudiante*, d'*histoire*, de *cours* ; ainsi que la relation *assister* et les contraintes que sont le type de cours auxquels peuvent assister les étudiants et les étudiantes, ceux prévus dans l'année, leur nom, etc. Les connaissances factuelles sont les faits que *Nouka* est une *étudiante*, que cette *étudiante assiste* au *cours 1H001* et que ce *cours* est une leçon d'*histoire*.

Formalisation

Formellement, le vocabulaire est défini comme un quintuplet :

$$\mathcal{V} = (T_C, T_R, \mathcal{M}, \sigma, \tau,)$$

dont les éléments sont décrits successivement ci-après.

T_C et T_R correspondent respectivement aux types de concept et aux types de relation : les premiers sont des prédicats de la logique des prédicats du premier ordre d'arité 1, les seconds des prédicats d'arité supérieure ou égale à 1. Ce sont des ensembles finis et partiellement ordonnés, dont la relation d'ordre est la subsomption.

En reprenant l'exemple illustratif, *Étudiante* et *Histoire* sont des éléments de T_C alors que *assister* est un élément de T_R. Nous introduisons également le concept *Cours* tel que l'ordre partiel sur T_C donne *Cours* comme plus général que *Histoire* ; *Étudiante* et les deux autres types sont incomparables. Cet exemple de T_C est représenté en figure 1.5, page 27,

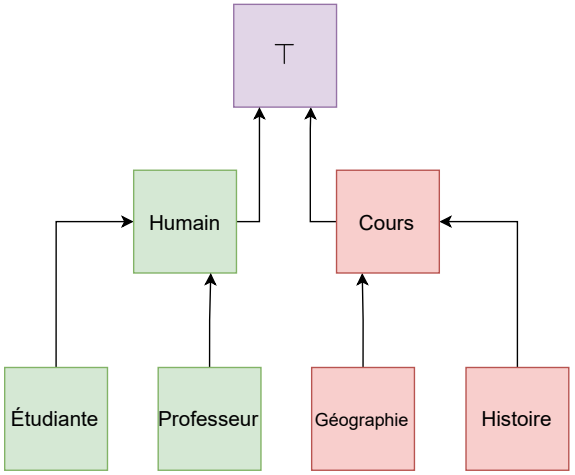


FIGURE 1.5 – Hiérarchie de concepts pour l'exemple illustratif

où $\top$ est le type maximal par défaut dans une hiérarchie de concepts dans un vocabulaire. Pour l'exemple des attaques informatiques, T_C est représenté sur la figure 1.3 déjà introduite, et la figure 1.4 est un exemple de T_R réduit aux types de relations d'arité 2.

Chaque type de relation a une arité fixe : T_R peut être décomposé en une partition selon ces arités. Par exemple, *assister* est un type de relation d'arité 2. Chaque relation est également associée à une signature, via la fonction σ, qui spécifie les types de concept maximaux pour chaque argument de la relation : formellement, pour tout $t \in T_R$, $\sigma(t)$ est un n-uplet de T_C, où $n \in \mathbb{N}$ est l'arité de t. Par exemple, *assister* a pour signature *(Étudiante, Cours)* car les instances mises en jeu par cette relation sont de type respectivement *Étudiante* et *Cours*, ou une de leurs spécialisations. Pour le cas des attaques, la figure 1.6, page 28 représente les signatures des relations d'arité 2. Les entiers représentés sur les arcs indiquent les positions dans le n-uplet. Des relations ternaires introduites plus loin dans le chapitre sont associées aux signatures données dans la figure 1.9, page 31.

$\mathcal{M} = I \cup \{*\}$ est un ensemble de marqueurs utilisés pour identifier les instances des nœuds conceptuels : I est l'ensemble des marqueurs individuels qui identifient une entité spécifique de l'univers considéré et $*$ est le marqueur générique qui fait référence à une entité non spécifiée. Par exemple, *Nouka* et *1H001* sont des entités spécifiques de I.

Enfin, τ est une fonction de conformité définie de I vers T_C, et qui définit le type de concept le plus spécifique instancié par chaque marqueur individuel. Par exemple, $\tau(Nouka) = Étudiante$; $\tau(1H001) = Histoire$.

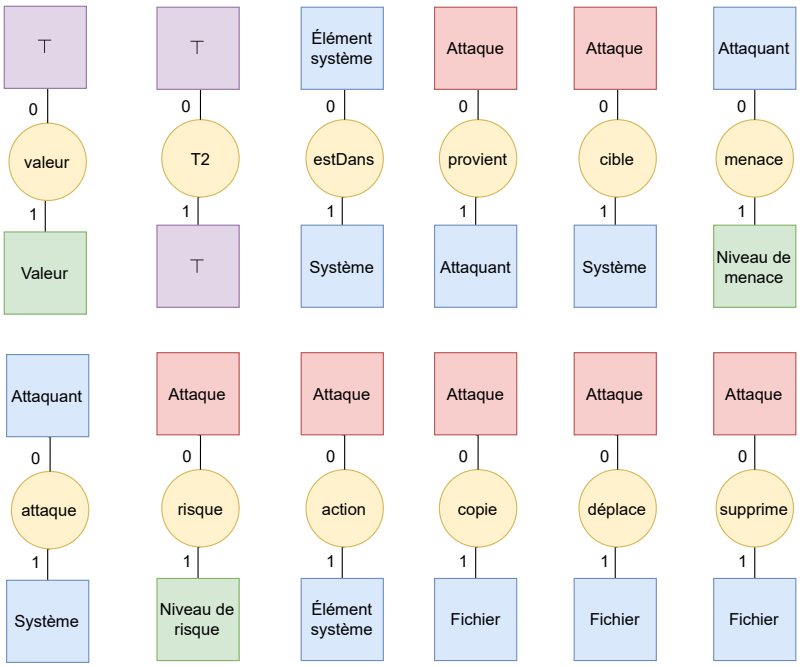


FIGURE 1.6 – Signatures des relations d'arité 2 utilisées pour représenter des attaques informatiques

1.2.3 Connaissances factuelles : multigraphes étiquetés

Cette section est une formalisation de la représentation de connaissances factuelles dans les formalismes des graphes conceptuels. Les éléments définis dans le vocabulaire constituent le langage utilisé par la partie factuelle au sein de multigraphes étiquetés, comme représentation graphique de formules de la logique des prédicats du premier ordre.

Définition formelle

Un graphe conceptuel est un multigraphe étiqueté bipartite représenté par un quadruplet :

$$G = (C, R, E, label)$$

faisant référence au vocabulaire défini $\mathcal{V}$.

C et R correspondent respectivement aux nœuds concept et aux nœuds relation. Contrairement aux multigraphes classiques, les relations entre concepts ne sont pas représentées par des arêtes mais par des nœuds relation.

E est l'ensemble des arêtes reliant les éléments de C aux éléments de R, c'est-à-dire les nœuds concept aux nœuds relation.

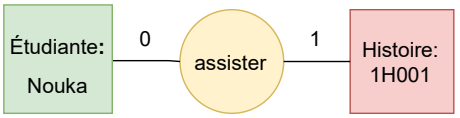


FIGURE 1.7 – Représentation de l'exemple illustratif par la notation graphique

label est une fonction d'étiquetage de C vers $T_C \times \mathcal{M}$, de R vers T_R et de E vers l'ensemble des entiers naturels $\mathbb{N}$. Pour tout $r \in R$, $label(r) = t_r \in T_R$ est le type de r et pour tout $c \in C$, $label(c) = (t_c, m_c) \in T_C \times \mathcal{M}$, où t_c est le type de c et m_c est le marqueur de c. Enfin, pour tout élément e de E, *label* renvoie, pour c et r respectivement le nœud concept et le nœud relation reliés par e, la position de c dans l'ordre des arguments de r.

Notations textuelle et graphique

L'exemple concernant l'étudiante *Nouka* peut être représenté par un graphe conceptuel comprenant deux nœuds concept, respectivement étiquetés par les couples suivants :

$$[\textit{Étudiante} : \textit{Nouka}]$$

$$[\textit{Histoire} : \textit{1H001}]$$

et un nœud relation connecté à chaque nœud concept, étiqueté *assister*.

La figure 1.7, page 29 représente cet exemple en utilisant une notation de graphe, avec trois nœuds n_1, n_2 et n_3 utilisant les étiquettes définies, avec :

$$label(n_1) = \textit{Étudiante} : \textit{Nouka}$$

$$label(n_2) = \textit{Histoire} : \textit{1H001}$$

$$label(n_3) = \textit{assister}$$

tandis que les arêtes sont étiquetées 0 et 1 pour spécifier le rôle respectif de chaque nœud concept dans la relation *assister*.

Il est à noter que dans la représentation graphique, par souci de concision, le marqueur générique n'est pas représenté : à la place, seul le type est renseigné.

La notation textuelle représente un nœud concept étiqueté par le type c et le marqueur m comme :

$$[c : m]$$

et un nœud relation étiqueté par le type r, connecté à deux nœuds concept respectivement étiquetés par $[c : m]$ et $[c' : m']$ par :

$$\begin{array}{ll} (r) & -0-[c : m] \\ & -1-[c' : m'] \end{array}$$

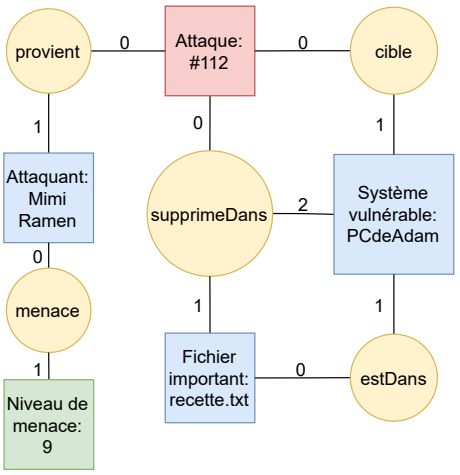


FIGURE 1.8 – Représentation d'un exemple plus complexe d'une attaque informatique par la notation graphique

où 0 et 1 sont les étiquettes des arcs identifiant le rôle des concepts $[c : m]$ et $[c' : m']$ au sein de la relation r.

Suivant ces notations textuelles, le fait correspondant à l'exemple considéré, représenté graphiquement sur la figure 1.7, est noté :

$$\begin{aligned}(assister) \quad & -0-[\acute{E}tudiante : Nouka] \\ & -1-[Histoire : 1H001]\end{aligned}$$

Enfin, un exemple plus complexe, dans le cadre des attaques informatiques, est donné figure 1.8, page 30. Il utilise les concepts représentés dans la figure 1.3, page 25, les relations binaires des figures 1.4, page 26 et 1.6, page 28 ainsi qu'un exemple de relation ternaire dont la signature est représentée dans la figure 1.9, page 31. Elle inclut un nœud concept particulier, comprenant une valeur, non introduit ici. C'est une extension des graphes conceptuels (CHEIN & MUGNIER, 2008), utilisée et décrite plus tard en détail dans le chapitre 2, section 2.2.1, page 42.

1.2.4 Règles-λ

Plusieurs extensions du cadre de base précédent des graphes conceptuels ont été proposées (CHEIN & MUGNIER, 2008) : elles comprennent par exemple la représentation de règles d'inférence et de contraintes (BAGET & MUGNIER, 2002), de valeurs numériques non symboliques (SOWA, 1983 ; THOMOPOULOS et al., 2003b), de données typées (BAGET, 2007 ; RAIMBAULT et al., 2009), de propriétés temporelles (KAMSU-FOGUEM et al., 2013), de graphes conceptuels imbriqués, ou bien *nested conceptual graphs* (CHEIN & MUGNIER,

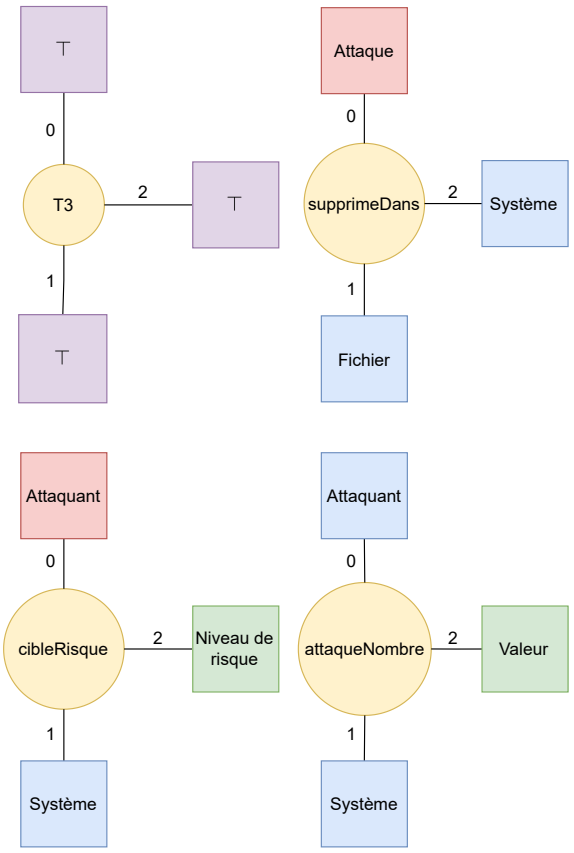


FIGURE 1.9 – Signatures de relations d'arité 3 pour l'exemple des attaques informatiques

1997), de graphes conceptuels stratifiés, ou *layered conceptual graphs* (CROITORU et al., 2005), les classes de coréférence (SOWA, 1983 ; CHEIN & MUGNIER, 2004), ainsi que les graphes conceptuels flous abordés dans le chapitre 2. Elles permettent d'accroître l'expressivité et l'interprétabilité des graphes conceptuels ou l'efficacité des outils les manipulant.

Nous présentons ici les règles-λ, qui déterminent une troisième sorte de connaissances en plus des connaissances ontologiques et factuelles : les connaissances implicites.

Elles consistent en des règles de la forme $R : H \longrightarrow C$, où H et C sont deux graphes conceptuels particuliers respectivement nommé *hypothèse* et *conclusion de la règle*. Leur application mène à l'extension des graphes conceptuels d'une base par l'ajout de nœuds supplémentaires, ou à la spécialisation de leurs étiquettes, comme détaillé ci-dessous.

La figure 1.10 représente un exemple avec une extension de graphe tandis que la figure 1.11, page 33 représente un exemple avec une spécialisation de graphe. Les types *fichierVulnérable* et *attaqueDangereuse* seraient à ajouter dans la hiérarchie représentée dans la figure 1.3, page 25 en spécialisation des concepts *fichier* et *attaque* respectivement.

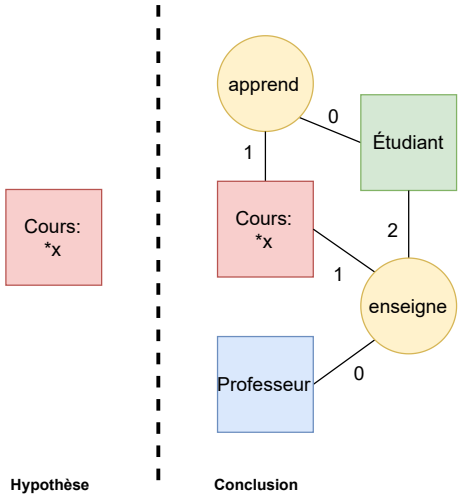


FIGURE 1.10 – Exemple de règle-λ entraînant une extension de graphe

Comme illustré dans ces exemples et évoqué plus haut, une règle λ est une paire ordonnée de graphes conceptuels-λ composée d'une hypothèse et d'une conclusion, où un graphe conceptuel-λ est un graphe conceptuel comportant des nœuds de correspondance définis. Ces derniers sont des nœuds concept de marqueur générique ayant chacun une variable associée utilisée. Les nœuds de correspondance de l'hypothèse sont associés à leur équivalent dans la conclusion.

Les figures 1.10, page 32 et 1.11, page 33 représentent deux nœuds de correspondance ayant la variable "$*x$" associée.

L'application d'une règle peut être l'extension d'un graphe conceptuel, la conclusion incluant alors l'hypothèse, ou la spécialisation d'un motif, la conclusion étant alors une copie de l'hypothèse avec des étiquettes plus spécifiques. Elle peut également combiner ces deux effets simultanément.

La règle de la figure 1.10 permet de compléter un graphe conceptuel comportant un nœud concept de type *Cours* en ajoutant qu'il est enseigné par un *Professeur* et suivi par (au moins) un étudiant. La règle de la figure 1.11 permet également de compléter un graphe conceptuel en indiquant que si un fichier y est supprimé par une attaque x, alors y est plus précisément de type *fichier vulnérable* et x de type *attaque dangereuse*. Dans les deux exemples, les règles-λ représentent des connaissances implicites des experts du domaine, qui permettent d'enrichir automatiquement les connaissances représentées par un graphe.

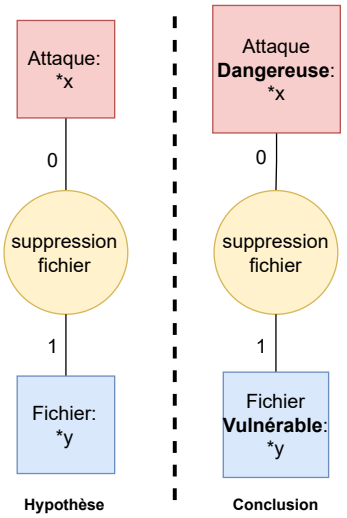


FIGURE 1.11 – Exemple de règle-λ entraînant une spécialisation de graphe

1.2.5 Outils de manipulation

Le chapitre s'est focalisé sur la représentation de la connaissance, mais les graphes conceptuels sont également munis d'outils de manipulation, permettant de les exploiter, et justifiant le choix de cette représentation. De plus, les exploitations ou manipulations logiques peuvent être réalisées par le biais d'outils issus de la théorie des graphes, qui les rendent très efficaces. Nous illustrons l'un de ces outils pour les raisonnements dans les graphes conceptuels, la subsomption (CHEIN & MUGNIER, 2008).

La *subsomption* est une relation entre deux connaissances. On dit qu'une connaissance A subsume une connaissance B, ou que B est subsumé par A, si B implique A, ou encore si A est une généralisation de B. L'idée est que A décrit alors une connaissance moins précise qui est déductible de la connaissance de B.

La relation de *généralisation* notée $\geq$ est définie par : étant donné deux graphes G et H, G spécialise H, c'est-à-dire $H \geq G$, si H peut être obtenu à partir de G par une séquence d'opérations simples sur le graphe. Ces opérations sont, nœud à nœud, basée sur les ordres dans T_C et T_R du vocabulaire, et sur la relation $* \geq i$ pour tout i de I.

La définition est rappelée ici pour les cas successifs de graphes génériques, graphes étiquetés, graphes étiquetés avec taxonomie et enfin graphes conceptuels.

Un graphe conceptuel en subsume un autre s'il existe un *homomorphisme* du premier vers le second. L'homomorphisme, également appelé projection, est une fonction qui peut être définie comme faisant correspondre nœud à nœud deux graphes, où un nœud du premier graphe a un type plus général que son nœud correspondant dans le second graphe, et en préservant la connexité : deux nœuds connexes dans le graphe de

départ doivent voir leurs images dans le graphe d'arrivée également connectées. La présence d'une relation d'homomorphisme entre deux graphes traduit que l'information contenue dans le premier est présente dans le second.

Ainsi, soient $G = (V, E)$ et $H = (W, F)$ deux graphes génériques, sans étiquettes, où V et W sont deux ensembles de nœuds, et E et F sont deux ensembles d'arcs tels que : $E \subseteq \{\{a, b\} \in V^2\}$; $F \subseteq \{\{a, b\} \in W^2\}$. Un homomorphisme entre G et H peut être défini comme une fonction $\pi : V \rightarrow W$ telle que :

$$\forall \{a, b\} \in E, \{\pi(a), \pi(b)\} \in F$$

Dans le cas de graphes étiquetés, on peut exiger l'égalité des étiquettes des nœuds mis en correspondance. La fonction π devient alors :

$$\forall \{a, b\} \in E, \{\pi(a), \pi(b)\} \in F \wedge a = \pi(a) \wedge b = \pi(b)$$

On admet qu'un nœud est ici confondu avec son étiquette.

Dans le cas où une relation d'ordre est définie sur les étiquettes, et que cette relation implique la subsomption $\geq$, on peut assouplir la mise en correspondance des nœuds afin qu'une telle fonction π mette en correspondance un nœud à son image si cette image est subsumée par le nœud antécédent. On obtient ainsi pour π :

$$\forall \{a, b\} \in E, \{\pi(a), \pi(b)\} \in F \wedge a \geq \pi(a) \wedge b \geq \pi(b)$$

Enfin, dans le cas des graphes conceptuels, on peut vouloir ajouter que la mise en correspondance se fait nœud concept à nœud concept et nœud relation à nœud relation.

Une multitude d'homomorphismes différents peuvent ainsi être définis selon les conditions fixées (Hell & Nesetril, 2004 ; Chein & Mugnier, 2008 ; Naserasr et al., 2015). Par exemple, dans le cas des graphes conceptuels, on peut vouloir ne considérer que les nœuds sans prendre en compte la connexité, ou bien ne se limiter qu'aux nœuds concept par exemple.

Cette notion d'homomorphisme est à la base des règles-λ de la section 1.2.4, page 30, des opérations de traduction entre différents formalismes du chapitre 3, page 77, de l'opérateur de fusion ainsi que de la correspondance entre graphe conceptuels-γ et les bases générées du chapitre 4, page 97 ou encore de la notion de motifs fréquents du chapitre 5, page 119. Chacun de ces travaux utilise une notion différente d'homomorphisme qui pourrait être explicitée.

1.3 Bilan

Ce chapitre a présenté les formalismes utilisés dans la suite de la thèse, pour la représentation de connaissances imprécises et structurées respectivement.

Les graphes conceptuels ont été conçus avec l'idée de disposer d'une représentation des connaissances claire pour un humain. L'objectif visé a été de permettre un certain niveau d'interprétabilité des connaissances factuelles ainsi représentées, des connaissances ontologiques sur lesquelles elles reposent et des différentes sortes de connaissances, par l'homogénéité du format imposé : les graphes étiquetés. Les raisonnements s'effectuent au moyen d'opérations de graphes dont les étapes sont représentables visuellement et rendent explicites les modifications ainsi effectuées, permettant d'assurer un niveau d'explicabilité également.

Par ailleurs, comme évoqué en section 1.2.4, page 30, il existe de nombreuses extensions des graphes conceptuels dont les graphes conceptuels flous discutés dans le chapitre 2, page 37 et notre proposition de graphes conceptuels-γ du chapitre 4, page 97.

Enfin, le formalisme des graphes conceptuels est également mis en œuvre au travers de l'exploitation de bases de graphes conceptuels en partie III, page 117 pour notre proposition d'algorithme *cgSpan*, pour l'extraction de motifs fréquents non-redondants avec les connaissances ontologiques, présenté dans le chapitre 5, page 119, dont le chapitre 6, page 139 présente une application à des données réelles.

Chapitre 2

Connaissances structurées imparfaites : étude comparative des graphes conceptuels flous

Les graphes conceptuels flous constituent une extension des graphes conceptuels, qui exploitent la logique floue de manière à accroître à la fois l'expressivité et l'interprétabilité des formalismes de la famille des graphes conceptuels.

Les graphes conceptuels flous permettent en effet un certain degré d'expressivité, car l'on passe alors de la représentation de connaissances binaires, où chaque élément est soit vrai soit faux, sans nuance possible, à la représentation de connaissances imparfaites, plus proches de la réalité et de ses modalités variées.

La figure 2.1, page 38 reprend l'exemple de *Nouka* la figure 1.7, page 29 en représentant cette fois un cours de type $(Histoire, 0.7) \wedge (Géographie, 0.3)$, qui représenterait un cours qui traiterait majoritairement d'histoire, mais aussi partiellement de géographie. Ainsi cet exemple illustre un cas de type flou. Dans le cas des attaques informatique, la figure 2.2, page 38 adapte la figure 1.8, page 30 : des exemples de type flou, de valeur floue, de marqueur flou et de relation floue sont représentés. Les pondérations qui figurent dans ce graphe sont présentées puis discutées dans ce chapitre.

Les graphes conceptuels assurent par ailleurs un niveau d'interprétabilité car la représentation de telles connaissances imparfaites est plus proche du langage naturel. L'utilisation de symboles comme les termes *Haut* et *assezVrai* dans la figure 2.2, page 38, permet une meilleure compréhension intuitive par un humain de la connaissance représentée. Par exemple, un système d'information peut manipuler ces termes flous pour renvoyer des résultats plus fins. En effet, au lieu de devoir réagir à *attaqueDangereuse* uniquement, disposer d'une gradualité par l'utilisation d'$(attaqueDangereuse, v_{danger})$ où v_{danger}

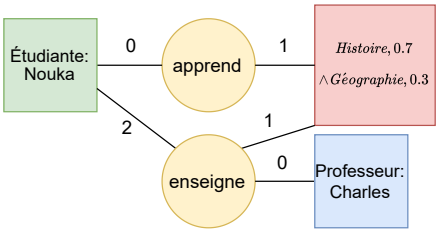


FIGURE 2.1 – Exemple de graphe conceptuel comportant une conjonction pondérée de types

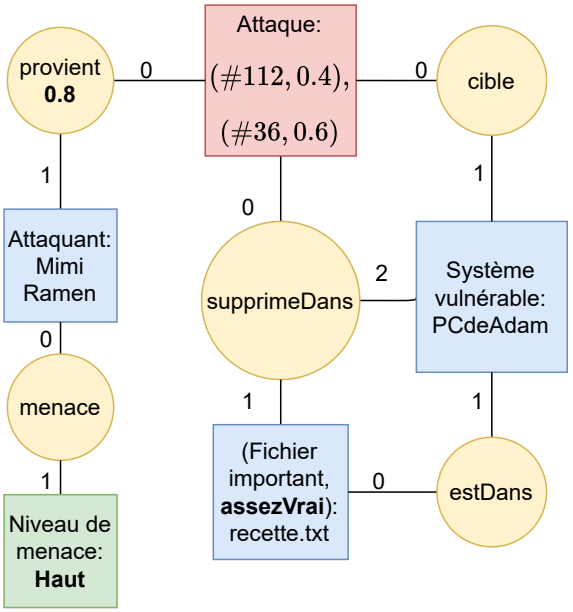


FIGURE 2.2 – Exemple de graphe conceptuel flou pour les attaques informatiques

traduit par une valeur dans $[0,1]$ le degré de vérité de *attaqueDangereuse*, permet à un système d'information d'adopter des comportements en fonction de v_{danger}.

Il existe plusieurs propositions de l'état de l'art pour l'extension de graphes conceptuels en graphes conceptuels flous. Ces propositions élargissent la capacité de représentation des connaissances des graphes conceptuels par la définition d'un poids ou l'intégration d'ensembles flous, qui quantifient l'imprécision. Nous proposons dans ce chapitre une étude comparative des formalismes existants, en nous concentrant sur la capacité de représentation des graphes conceptuels flous plutôt que sur les raisonnements associés.

La section 2.1 présente les différents axes de discussion au prisme desquels les formalismes flous sont étudiés. Les sections suivantes identifient et discutent différentes extensions floues des graphes conceptuels. Les trois premières sections traitent de dif-

férents cas d'extensions floues dans des nœuds concept : la section 2.2, page 41 détaille les propositions qui enrichissent les nœuds concept avec un coefficient de pondération ; la section 2.3, page 56 discute le cas des nœuds concept avec attribut flou ; la section 2.4, page 61 examine les nœuds concept admettant une pondération conjonctive ou disjonctive de plusieurs types pour un même nœud. La section 2.5, page 68 enfin discute des contributions sur les extensions floues simples ou les transpositions des cas précédents.

Le travail présenté dans ce chapitre a donné lieu à une publication dans les actes de la conférence SSCI-FOCI 2021 (Faci et al., 2021c).

2.1 Présentation générale

Cette section présente d'abord les principes et motivations de la discussion comparative que nous proposons sur les modèles de graphes conceptuels flous, en section 2.1.1. Elle présente ensuite la grille de lecture que nous proposons pour détailler par quel prisme les articles sont étudiés en section 2.1.2, page 40. Nous poursuivons avec deux taxonomies proposées pour catégoriser les modèles et leurs interprétations respectives : la première offre une synthèse de ces modèles avec, pour chacun, une définition succinte et un exemple, tandis que la seconde donne une vision globale des modèles de graphes conceptuels flous dans ce chapitre, fournissant la partie des graphes conceptuels impactée et la section qui détaille le modèle dans ce chapitre.

2.1.1 Principes et motivations

L'objectif de ce chapitre est de proposer un état de l'art des différentes méthodes utilisées pour représenter les informations imprécises dans les graphes conceptuels afin d'étendre leur capacité de représentation des connaissances.

Applications

Ces propositions ont été exploitées dans différents domaines d'application dont nous citons ici quelques exemples.

Dans le contexte de la modélisation d'images, Mulhem et al., 2001 proposent de comparer des images avec des modèles de scènes naturelles. Une image y est représentée par un graphe conceptuel flou comprenant des degrés de confiance relatifs à la classe d'une région de l'image, fournis par un algorithme. Un modèle de scène naturelle est représenté par un graphe conceptuel flou utilisant des degrés de vérité pour représenter l'intensité de relations entre régions de la scène, tel que *est plus petit que* ou *est en dessous de*. Cet article propose également un algorithme pour la comparaison de graphes conceptuels

flous par une recherche d'homomorphisme entre le graphe conceptuel flou d'une image et le graphe conceptuel flou d'un modèle de scène naturelle.

Dans le contexte de l'interrogation de bases de données, Thomopoulos et al., 2006 proposent d'utiliser leur proposition d'ensembles flous hiérarchiques, afin d'obtenir des résultats plus importants pour une même requête. Seule la hiérarchie sur les types y est floue. Une requête est formée d'un graphe conceptuel flous dont les valeurs de vérité sont propagées aux types plus généraux ou plus spécialisés que ceux utilisés dans le graphe.

Enfin, dans le contexte de la mise en correspondance d'ontologies, Buche et al., 2008 proposent d'utiliser des graphes conceptuels flous afin de représenter une ontologie où les éléments flous permettent une représentation des similarités entre objets et entre valeurs des deux ontologies. Ainsi, des objets et valeurs qui n'auraient pas pu être mis en correspondance dans le cadre non-flou peuvent l'être ici par les graphes conceptuels flous, et une fonction de similarité entre les ontologies se base sur ces graphes.

2.1.2 Grille de lecture

Tout au long du chapitre, chaque proposition intégrant des composantes floues est d'abord présentée par sa définition. Il s'agit d'une transposition de l'article original adaptée à la notation introduite dans le chapitre 1. Une formalisation utilisant la notation textuelle suit afin de résumer la proposition.

L'étude comparative que nous proposons discute des approches existantes sur leurs avantages respectifs et leurs éventuelles limitations. La discussion s'articule autour de trois axes : (a) Vue critique de chaque approche et comparaison avec les propositions précédentes de l'état de l'art ; (b) Présentation des interprétations possibles de la définition donnée pour illustrer son potentiel et ses limites ; (c) Clarification de la partie des graphes conceptuels impactée par la définition ainsi que la contrainte relâchée.

Nous parlons ici de graphes conceptuels flous, mais selon le modèle, l'interprétation en tant que connaissance floue n'est pas toujours explicite : elle y est parfois confondue ou remplacée par, par exemple, une interprétation possibiliste.

2.1.3 Taxonomies

Le tableau 2.1, page 42 résume notre proposition de structuration des modèles de graphes conceptuels flous selon la partie du formalisme qui est concernée et la section qui traite de son cas. Ces modèles sont interprétés de plusieurs façons dans ce manuscrit : plusieurs interprétations sont fournies pour tenter de combler l'ambiguïté des modèles étudiés.

Les tableaux 2.2, page 43, 2.3, page 57, 2.4, page 62 et 2.5, page 67 sont en début des sections qui les traitent. Ils présentent notre seconde proposition de taxonomie qui ren-

seigne l'ensemble des modèles présentés dans ce chapitre avec plus de détail. Pour chacun des modèles, une synthèse des définitions fournies par les articles (deuxième colonne) et un exemple illustratif (troisième colonne) y sont associés en reprenant celui de *Nouka* introduit en chapitre 1. Il est à noter que les interprétations des exemples illustratifs sont des suggestions, et que des interprétations alternatives sont possibles, les articles ne se limitant que rarement à une seule. Les codes de l'interprétation correspondante dans le tableau 2.1, page 42 ainsi que dans leur présentation dans les sections correspondantes sont renseignés avant chaque exemple et son interprétation. L'objectif de ces propositions est d'offrir une synthèse résumant l'essentiel des propositions étudiées. Les notations utilisées reprennent le cas d'un vocabulaire $\mathcal{V} = (T_C, T_R, \mathcal{M}, \sigma, \tau,)$ et d'un graphe conceptuel $G = (C, R, E, label)$.

Une première séparation est effectuée selon que la partie ontologique ou la partie factuelle est affectée par le modèle. Une distinction plus fine est effectuée sur la première colonne selon l'élément dont les contraintes de représentation sont relâchées. Il est à noter que les lignes indiquant *Global* recensent les modèles où une distinction plus fine n'a pas pu être opérée, par exemple si le modèle concerne l'ensemble des connaissances ontologiques (resp. factuelles). La section du chapitre où la proposition est introduite est indiquée dans la deuxième colonne. La troisième colonne contient un code qui différencie les interprétations selon la spécification renseignée en fin de section 2.1.1, page 39. Si des interprétations plus fines ne changent pas l'emplacement des connaissances concernées, comme c'est le cas avec $(2a)$ qui est affiné en $(2a1)$ et $(2a2)$ dans sa section 2.2.1, page 42, le code le plus général est renseigné sans plus de précision. La quatrième colonne enfin fournit les références du ou des articles correspondants.

2.2 Nœud conceptuel pondéré

Une première famille de méthodes d'extension des graphes conceptuels en graphes conceptuels flous propose de représenter les connaissances imprécises en enrichissant les nœuds concept. [1] En particulier, les modèles discutés quantifient l'imperfection de la connaissance par un poids associé au nœud concept. Les contraintes du modèle proposé ainsi que son interprétation varient selon les articles. Il est cependant à noter que dans les trois propositions qui suivent, les interprétations possibles ne sont données, par les articles, qu'à titre indicatif afin d'illustrer et motiver la proposition.

Ainsi, afin de quantifier l'imperfection, le premier modèle propose l'utilisation d'une valeur numérique comprise dans $[0, 1]$. Le deuxième modèle propose d'utiliser une va-

1. Les travaux de Morton, 1987 n'étant pas accessibles, ils sont étudiés à travers leur transcription dans l'article de Wuwongse et Manzano, 1993.

	Emplacement	Section	Code	Référence
Ontologique	**Global**	2.2.1, page 42	(2b)	Morton, 1987
		2.4.3, page 65	(7b)	Thomopoulos et al., 2003a
		2.5.3, page 70	(13)	Wuwongse et Cao, 1996 ; Cao et Creasy, 2000
	Type de concept	2.2.3, page 54	(4)	Cao et al., 1997 ; Cao, 2010
		2.4.2, page 64	(7a)	Thomopoulos et al., 2003b
		2.4.1, page 63	(6)	Cao, 1999
	Type de relation	2.5.1, page 68	(8)	Wuwongse et Manzano, 1993
			(9)	Wuwongse et Cao, 1996
			(10)	Cao et al., 1997
	Marqueur individuel	2.3.3, page 60	(5b)	Thomopoulos et al., 2003b
	Valeur	2.3.2, page 58	(5a)	Morton, 1987 ; Wuwongse et Manzano, 1993
Factuel	**Global**	2.5.2, page 69	(12)	Morton, 1987
	Concept	2.2.1, page 42	(2a)	Morton, 1987
		2.2.2, page 50	(3)	Wuwongse et Cao, 1996

Table 2.1 – Taxonomie proposée de l'emplacement des connaissances imprécises selon le modèle

riable linguistique typée selon son interprétation, telle que *trèsVrai* de type *Vrai*, qui se réfère à un sous-ensemble flou sur $[0,1]$. Le troisième modèle propose de définir au niveau du vocabulaire un treillis de types flous T_C^f, en tant que couples constitués d'un type de concept non flou de T_C et d'une variable linguistique. Le tableau 2.2 donne une vue synthétique de ces trois variantes.

2.2.1 Pondération par une valeur numérique

2.2.1.1 Syntaxe

Morton, 1987 propose de représenter l'imperfection de la connaissance d'un nœud concept avec marqueur individuel en l'associant à une valeur comprise entre 0 et 1.

Le nœud prend la forme :

$$[c : i, \alpha] \tag{2.1}$$

où c est un type de concept dans T_C, i est un marqueur individuel dans I et α est une valeur numérique de $[0,1]$.

D'après Wuwongse et Manzano, 1993, Morton, 1987 propose d'utiliser une fonction *partielle* dite de compatibilité (évoquée comme étant similaire à la définition de Sowa, 2008 des fonctions *type* et *referent*) qui, pour un nœud avec marqueur individuel i, I un

TABLE 2.2 – Nœud conceptuel pondéré

Modèle	Définition	Exemple et interprétation
Nœud conceptuel pondéré (localement ou globalement) par une valeur numérique (MORTON, 1987) 2.2.1, page 42	Représente les limites d'un processus perceptif à percevoir le monde extérieur Fonction de compatibilité : $\forall\ k \in\ C, \mu_k : T_C \times I \longrightarrow [0,1]$ Correspond à la compatibilité entre l'entité observée et le type attribué par le processus perceptif Modèle sémantique M définissant les fonctions de compatibilité globalement : elles ne dépendent plus d'un nœud concept de C	**(2a) :** $k = [Histoire : 1H202, \mu_k]$; avec $\mu_k(Histoire, 1H202) = 0.7$ peut signifier *À ma connaissance, j'ai comptabilisé que 70% du cours* 1H202 *traite d'*Histoire **(2b) :** $k = [Histoire : 1H202, \mu_k]$; avec $M(Histoire, 1H202) = 0.7$ peut signifier *70% du cours* 1H202 *traite d'*Histoire
Nœud conceptuel pondéré par une variable linguistique (WUWONGSE & CAO, 1996) 2.2.2, page 50	Modèle similaire au cas de la pondération numérique Trois différences : * Variable linguistique comme sous-ensemble flou dans $[0,1]$ au lieu d'une fonction de compatibilité * **(3a) :**Trois modalités de la variable dans un ensemble K prédéfinis de sous-ensembles flous dans $[0,1]$ munis de modificateurs linguistiques **(3b) :** Potentiellement une infinité de modalités renseignées dans K **(3c) :** Modalité non pas de la variable, mais inhérente à un couple de $T_C \times M$ * Variable typée selon une partition $K = \{True, False, Unknown\}$	$[Histoire : 1H202, plutôtVrai]$ peut signifier *Il est plutôt vrai que* 1H202 *traite d'*Histoire
Nœud conceptuel pondéré globalement (CAO et al., 1997 ; CAO, 2010) 2.2.3, page 54	Treillis de types flous T_C^f $\forall\ t^f \in\ T_C^f, \exists\ t \in\ T_C, \exists\ v \in\ K, t^f = (t, v)$ Dans CAO, 2010 T_C^f devient un ensemble muni d'une relation d'ordre partiel	$[Histoire_{très Vrai} : 1H202]$ peut s'interpréter comme le fait que *Le cours* 1H202 *est très majoritairement un cours d'*Histoire.

ensemble de marqueurs individuels et L_e un sous-ensemble de types du supertype *Entité*, fait correspondre une valeur dans $[0,1]$. Ainsi pour un nœud donné, on considère une fonction non pas sur l'ensemble des types *Entité*, mais une sous-partie de celui-ci.

D'autre part, la fonction étant partielle, elle n'est définie que sur une partie de L_e. Enfin, il est précisé qu'un nœud demeure constitué d'un unique type de concept et d'un unique marqueur individuel dans ce modèle, comme dans le cas classique, associé cependant à la fonction de compatibilité ainsi définie. Beaucoup d'éléments semblent ainsi définis sans explicitation de leur motivation.

Une telle pondération n'est pas définie pour un nœud concept étiqueté par le marqueur générique $*$ d'après Wuwongse et Manzano, 1993. En effet, la valeur α est donnée par une fonction dite de compatibilité définie pour un nœud concept donné, avec pour ensemble de définition $T_C^p \times I$ où T_C^p est une partie de T_C.

Dans cette proposition, Morton, 1987 traite des concepts instanciés par des marqueurs individuels. D'après Wuwongse et Manzano, 1993, ce modèle de flou perceptif (*perceptual fuzziness*) décrit la compatibilité entre une entité du monde réel et un type de concept, au sein d'un nœud concept avec marqueur individuel. Cette représentation est censée traduire les limitations d'un processus perceptif (*perceptual process*) qui associe des phénomènes à des représentations mentales d'entités physiques et abstraites. Plus généralement, ce modèle exprime la correspondance partielle qui s'opère à l'interface entre un processus perceptif et le monde extérieur.

Deux définitions de la compatibilité sont données, respectivement pour la forme graphique et la forme logique. La première consiste en la définition d'une fonction partielle $\mu_k : T_C^p \times I \longrightarrow [0,1]$, où T_C^p est une partie de T_C, pour chaque nœud concept individuel $k \in C$. Cette définition correspond à la présentation qui est faite du modèle jusqu'ici dans cette section, et correspond à la formalisation d'une compatibilité de portée locale au nœud concept.

La seconde consiste en la définition de $M : T_C \times I \longrightarrow [0,1]$ intégrée dans le vocabulaire, une fonction définie pour tout graphe conceptuel flou défini sur ce vocabulaire. Cette définition correspond à une compatibilité globale, car la fonction de compatibilité M n'est alors plus définie pour chacun des nœuds concept flous avec marqueur individuel. D'autre part, elle n'est plus définie sur une partie T_C^p de T_C conjuguée à I, mais sur $T_C \times I$.

Ainsi, la motivation principale de ce modèle est de créer une nouvelle possibilité en terme de représentation de la connaissance, menant à un accroissement de l'expressivité des nœuds concept.

2.2.1.2 Exemple illustratif

En reprenant le cas de *Nouka*, on peut avoir :

$$[\textit{Histoire} : \textit{1H202}, 0.7]$$

La valeur peut représenter l'imprécision inhérente à un type, par exemple, Histoire *couvre de nombreux éléments qui ne sont pas clairement définis, il est donc difficile de dire que le cours* 1H202 *est un cours d'*histoire, ou la difficulté de déterminer précisément sa compatibilité avec l'entité considérée, par exemple, *Le cours* 1H202 *est un cours d'*histoire, *mais contient également des parties de* géographie.

Une autre interprétation peut être proposée en considérant justement les limitations du processus perceptif qui n'a pas pu capter ou traiter toutes les informations issues du cours *1H202*, à cause de ses limites mais également des perturbations lors de la capture (bavardages, interruption, …). Il est d'ailleurs en soi impossible de capter et traiter l'infinité d'informations associées à une entité, et donc de parfaitement la définir. On peut ainsi interpréter la valeur 0.7 comme une estimation de l'imperfection avec laquelle on décrit le cours.

2.2.1.3 Interprétation

L'inclusion des connaissances imparfaites est effectuée à un niveau différent selon la définition considérée. Ainsi, selon Wuwongse et Manzano, 1993, comme détaillé ci-après, Morton, 1987 définit soit α comme étant spécifique à chaque concept individuel ($2a$) via la fonction de compatibilité μ_k, donc localement, soit comme étant définie au niveau du vocabulaire ($2b$) via la fonction M, et donc globalement. Ces deux définitions sont interprétées et discutées tour à tour dans ce qui suit.

Pondération locale (2a)

Définition La définition ($2a$) établit la fonction de compatibilité locale :

$$\mu_k : T_C^p \times I \longrightarrow [0,1]$$

qui correspond à la fonction caractéristique d'un sous-ensemble flou sur $T_C^p \times I$. On peut également définir ce sous-ensemble flou sur $T_C \times I$ en associant 0 aux éléments membres du complémentaire de T_C^p dans T_C. Cette fonction caractéristique est spécifique à chaque nœud de chaque graphe conceptuel flou d'une base de connaissances.

Exemple On peut ainsi définir par exemple :

$$a = [Histoire : \ 1H202, \ \mu_a]$$

$$b = [Histoire : \ 1H202, \ \mu_b]$$

où $\mu_a(Histoire, 1H202) = 0.7$ et $\mu_b(Histoire, 1H202) = 0.4$. Dans ce contexte, on voit bien que la compatibilité n'a pas du tout la même sémantique que celle d'une valeur inhérente au couple *(type,individu)*.

Un même couple $(t, i) \in \ T_C \times \ I$ a ainsi potentiellement une valeur de compatibilité différente d'un nœud concept individuel à l'autre.

Contraintes relâchées Deux contraintes sont ainsi relâchées dans le formalisme des graphes conceptuels par la définition de ce modèle $(2a)$ de graphe conceptuel flou : la première est que la valeur de compatibilité entre un type et un individu au sein d'un nœud concept devient floue, son domaine de valeur passant de $\{0, 1\}$ à $[0, 1]$; le second relâchement de contrainte est le fait que cette valeur n'est pas unique pour un couple donné.

Cette proposition peut être modérée si l'on considère l'unicité d'occurrence d'un marqueur individuel dans un graphe conceptuel flou, voire dans une base de connaissance. Le premier cas correspond à une base de graphes conceptuels flous normaux, qui imposent que pour deux nœuds concept individuels distincts leurs marqueurs sont nécessairement différents. Différentes occurrences d'un même couple (t, i) peuvent néanmoins advenir au sein de la base de connaissances. Le second cas est la définition d'une base de graphes conceptuels généralisant la contrainte de normalité à tous les graphes. Un même couple (t, i), et a fortiori le marqueur i, ne peut pas apparaître plus d'une fois.

Interprétations Une première interprétation possible en accord avec la définition informelle de Morton, 1987 est que le processus perceptif change ses conclusions sur les caractéristiques de *1H202* entre deux captures. Ce changement de conclusion peut être dû, par exemple, à l'imprécision de ce processus lors de la capture, ou bien à l'apparition de nouvelles informations, ou encore, au caractère possiblement non-statique des limitations du processus perceptif qui évoluent au cours du temps.

Pour chaque concept individuel, le processus perceptif a une précision sur son observation en fonction du contexte (ou d'autres paramètres, intrinsèques ou extrinsèques). Cette précision peut alors varier pour une même entité donnée d'une capture à l'autre, et est évaluée par le processus via une fonction d'agrégation des connaissances imparfaites et des paramètres de capture.

Une seconde interprétation peut être qu'il n'y a aucun relâchement de contrainte, et que les graphes conceptuels qui en résultent ne sont pas des graphes conceptuels flous

mais des graphes conceptuels incertains. En effet, l'interprétation de cette valeur en tant que degré de certitude est proposée par Wuwongse et Manzano, 1993 dans leur présentation des travaux de Morton, 1987. D'autre part cela est en accord avec la définition informelle, et notamment avec *la correspondance partielle qui s'opère à l'interface entre un processus perceptif et le monde extérieur* : cette correspondance partielle mènerait ainsi à une incertitude sur les informations capturées par le processus.

Quelques éléments favorisent plutôt la première interprétation. L'objectif clairement établi par l'article (Morton, 1987 ; Wuwongse & Manzano, 1993) est d'obtenir un modèle de graphes conceptuels flous ; il y est fait référence aux travaux en logique floue ; et la nomenclature choisie pour les définitions est plutôt à propos d'imperfection que d'incertitude des connaissances.

Explications Pour la suite de cette partie, nous conservons les interprétations défendant l'interprétation des valeurs en tant que degrés de vérité, et donc permettant de contribuer à un modèle de graphes conceptuels flous. Les contraintes relâchées sont l'**unicité** et la **binarité** de la valeur de compatibilité pour un couple type-entité.

Nous proposons deux explications au relâchement de contraintes. Selon la première, le graphe conceptuel flou conserve la contrainte classique d'unicité : un seul couple de $T_C \times I$ est associé à chaque nœud concept individuel ($2a1$). Dans ce cas, l'ensemble flou d'un nœud donné n'est défini que pour un seul couple de $T_C \times I$, ce qui rend la définition de μ_k excessive. Seule la contrainte de binarité est relâchée.

Dans la deuxième explication, nous proposons de considérer que la contrainte d'unicité est également relâchée, ce qui conduit à des graphes conceptuels flous dont les nœuds concept individuels sont des ensembles flous sur $T_C \times I$, modélisant l'imprécision à la fois sur le type de concept et sur le marqueur individuel ($2a2$).

Cette seconde lecture permet d'avoir par exemple un nœud [(*Histoire* : *1H202*, 0.7), (*Géographie* : *1H202*, 0.4)], qui peut représenter l'imprécision sur plusieurs paires type-entité. La section 2.4, page 61 traite plus spécifiquement de tels nœuds étiquetés par plusieurs paires de $T_C \times I$, et examine plusieurs définitions offrant diverses interprétations, notamment conjonctive et disjonctive.

Dans les deux cas, un sous-ensemble flou sur $T_C \times I$ est ainsi défini pour chaque élément de C de chaque graphe conceptuel flou (chaque C étant spécifique à un graphe conceptuel flou) par les valeurs renvoyées par les fonctions de compatibilité.

Pondération globale (2b)

Définition La définition ($2b$) propose une fonction globale $M : T_C \times I \longrightarrow [0,1]$ dans le vocabulaire des graphes conceptuels flous. Il en résulte la proposition suivante :

la valeur de compatibilité pour un couple donné de $T_C \times I$ reste identique au sein de toute une même base construite sur ce vocabulaire.

Ainsi, alors que dans la définition ($2a$) la compatibilité est définie pour le triplet (k, t, i) où $k \in C$ d'un graphe conceptuel flou donné, $i \in I, t \in T_C$, elle devient, dans la définition ($2b$), définie pour le triplet $(\mathcal{V}, t, i)$.

Exemple Par conséquent, il y a la garantie que, par exemple, pour toute occurrence d'un nœud $[Histoire : 1H202, \alpha]$ dans tout graphe conceptuel flou du même vocabulaire, la compatibilité α entre *Histoire* et *1H202* est toujours de même valeur.

Contrainte relâchée La seule contrainte relâchée est la valeur de compatibilité qui adopte des valeurs dans $[0, 1]$ plutôt que $\{0, 1\}$.

Interprétations Nous proposons d'interpréter cette proposition comme le fait que la compatibilité définie devient inhérente à chaque couple de $T_C \times I$, quel que soit le nœud $k \in C$ considéré, c'est-à-dire que la compatibilité est spécifique à la relation d'assignation entre l'entité dans la réalité et le type de concept. Cette interprétation se justifie par une analogie entre cette valeur de compatibilité et la notion de caractéristiques inhérentes : dans les deux cas les notions sont statiques et spécifiques à ce à quoi elles sont attachées. Dans l'exemple de *Nouka*, un cours d'histoire inclut des dates et des rapports sociaux ; elles ne sont pas remises en question lors de chaque observation (la notion de cours d'histoire n'évolue pas ici) et elles font partie du savoir ontologique (elles découlent d'une manière de modéliser le monde représenté).

Dans le contexte de l'interprétation que nous proposons, un couple de $T_C \times I$ a ainsi comme caractéristique inhérente cette valeur de compatibilité. Notre interprétation de la compatibilité en tant que caractéristique intrinsèque du couple type-entité peut alors renvoyer directement au caractère intrinsèquement vague d'un type.

Deux paradigmes de graphes conceptuels Deux paradigmes sont à observer afin de montrer que le caractère statique de la valeur de compatibilité a un sens et est en accord avec les formalismes des graphes conceptuels. Le premier paradigme considère que le vocabulaire est établi avant la création des graphes conceptuels flous, alors l'observation de chaque entité dans I ne produit pas de nouveau savoir concernant ces valeurs de compatibilité. Dans le cas contraire, le second paradigme statue que chaque nouvelle connaissance factuelle ajoutée à la base de connaissance considérée entraîne potentiellement une modification du vocabulaire.

Ce sont deux paradigmes qui motivent différents fonctionnements des graphes conceptuels pour assurer ces valeurs de compatibilité statiques.

La conformité dans le premier cas est assurée à l'ajout de chaque graphe conceptuel flou par la vérification de sa conformité avec le vocabulaire. Si un des graphes conceptuels flous ajouté se révèle non conforme, dans ce paradigme il peut être refusé ou modifié afin de le rendre conforme au vocabulaire. Une alternative plus stricte est de considérer que de tels graphes conceptuels non conformes ne peuvent même pas être construits, le vocabulaire n'offrant pas les éléments de construction nécessaire.

Dans le second paradigme, lorsqu'un nouveau graphe conceptuel flou ajouté n'est pas conforme avec le vocabulaire, plusieurs sous-cas se présentent. Si la non-conformité est la présence de références à des connaissances ontologiques qui n'existent pas, il est possible alors de les ajouter dans le vocabulaire. Si la non-conformité est due à des informations contredisant celles dans le vocabulaire, par exemple si une valeur de compatibilité pour un couple de $T_C \times I$ donné n'est pas la même dans le graphe conceptuel flou ajouté et dans celle du vocabulaire, alors pour rétablir la conformité, soit on modifie le nouveau graphe conceptuel flou, soit on modifie le vocabulaire ainsi que tous les graphes conceptuels flous utilisant les éléments de connaissance ainsi modifiés.

Dans tous les cas, quel que soit le paradigme, une telle interprétation est bien en accord avec le fonctionnement d'une base de graphes conceptuels et le caractère statique de la compatibilité a bien un sens dans ce contexte.

Explications Les différences avec le cas ($2a$) de pondération locale des nœuds concept ont plusieurs explications possibles : l'origine de l'information imparfaite, le contexte de capture de cette information ou son interprétation en connaissance imparfaite.

La première explication ($2b1$) exprime que les cas ($2a$) et ($2b$) sont différents parce que l'information imparfaite provient de causes différentes : dans le cas ($2a$), la connaissance imparfaite définie dans la partie factuelle du graphe conceptuel flou est justifiée par sa production par un processus perceptif imparfait. Dans le cas ($2b$) l'ancrage dans la partie ontologique est justifié par la compatibilité partielle inhérente à chaque couple type-entité. Cette explication traduit ainsi le fait que dans le cas ($2b$), il existe une valeur de compatibilité unique pour un couple type-entité donné, alors que dans le cas ($2a$), il existe une multitude de valeurs pour le même couple. Cette multiplicité de valeurs pour un même couple peut être justifiée par des captures étalées dans le temps, dont les paramètres d'observation varieraient, ou par la capture par différents processus perceptifs imparfaits, aux caractéristiques diverses.

Une seconde explication ($2b2$) est une différence d'interprétation des informations imparfaites recueillies. Ainsi, soit la compatibilité traduit une incertitude sur la connaissance représentée, comme le propose Morton, 1987 d'après Wuwongse et Manzano, 1993, ou un manque de confiance, soit son caractère vague, soit parce qu'il y a une im-

précision sur la connaissance représentée ou de niveau de confiance dans la connaissance.

Le cas (2*a*) décrit un poids qui peut varier entre les observations d'un couple type-entité donné, et on peut donc s'interroger sur la sémantique floue de cette définition en terme d'imprécision. Les éléments mis en avant dans cette section peuvent plutôt faire penser que (2*a*) est proche d'un cadre d'incertitude, comme défendu par les auteurs d'ailleurs, ou de niveau de confiance.

Le cas (2*b*) décrit un poids inhérent au couple type-entité. Il est conforme à l'interprétation de la connaissance imprécise ou vague, car il peut y avoir un niveau d'imprécision ou de flou associé à un couple type-entité donné, qui est spécifique à ce couple et ne varie pas au cours des différentes observations du même couple. En ce sens, il s'inscrit totalement dans le cadre flou.

2.2.2 Pondération par une modalité linguistique

2.2.2.1 Syntaxe

Wuwongse et Cao, 1996 proposent dans ce deuxième modèle de représenter l'imperfection de la connaissance par une modalité linguistique λ. Un nœud concept s'écrit alors :

$$[c : m, \lambda], \lambda \in K \tag{2.2}$$

où m est un marqueur individuel ou le marqueur générique dans M, et K est un ensemble de modalités linguistiques définissant le degré de vérité, par exemple :

$$K = \{\mathit{trèsVrai}, \mathit{vrai}, \mathit{plutôtVrai}, \mathit{peuVrai} \ldots\}$$

Chacune des modalités est associée à un sous-ensemble flou de $[0,1]$, soit $\mu_\lambda : [0,1] \longrightarrow [0,1]$, au lieu d'une valeur unique $\alpha \in [0,1]$ dans le modèle qui précède. Une autre différence entre ces modèles est que Wuwongse et Cao, 1996 n'imposent pas que le marqueur associé m soit individuel, et donc peut être le marqueur générique $*$. Ainsi un nœud conceptuel flou peut représenter la compatibilité entre un type et une entité indéterminée.

Ainsi le vocabulaire d'un graphe conceptuel flou est enrichi par K, dont les éléments deviennent des éléments supplémentaires d'expression pour construire des graphes conceptuels flous.

Selon Wuwongse et Cao, 1996, le choix d'utiliser des modalités linguistiques conduit à des résultats plus pertinents lorsqu'il est intégré dans l'opération d'homomorphisme. C'est la motivation principale du choix de sous-ensemble flou comme pondération. Cependant, leurs arguments semblent discutables et le problème invoqué peut être lié à des

différences d'interprétation par rapport au cadre de Morton, 1987, comme discuté dans les sections suivantes.

L'ensemble K est l'union de trois ensembles disjoints T, F et U se référant respectivement à *Vrai*, *Faux* et *Inconnu*. L'appartenance d'une modalité linguistique à T, F ou U assure certaines propriétés : les termes dans T sont associés à des fonctions d'appartenance croissantes sur $[0,1]$ et égales à 1 en 1 ; les termes dans F sont associés à des fonctions d'appartenance décroissantes sur $[0,1]$ et égales à 1 en 0.

Cette partition de K modifie les mécanismes de raisonnement. En effet, elle implique que l'inférence entre types de concept associés aux termes de K n'est pas directement dérivée des principes classiques d'inférence entre types de concept de base et entre sous-ensemble flous.

L'homomorphisme proposé par Wuwongse et Cao, 1996 entre deux nœuds concept flous, nommé projection dans l'article, compare deux à deux leurs marqueurs, leurs types et leurs modalités et prend en compte la classe à laquelle appartient leur modalité.

Par ailleurs, ce modèle interprète les modalités comme la compatibilité entre le marqueur et le type au sein d'un nœud concept, de façon similaire au précédent modèle.

Enfin, une motivation de ce modèle pour l'utilisation d'une modalité linguistique au lieu d'une valeur numérique est de faciliter la manipulation de l'imprécision par les humains en la rendant plus intuitive, suivant un principe habituel en logique floue. En effet, l'idée est que la manipulation par un humain d'un terme tiré du langage naturel est plus confortable.

2.2.2.2 Exemple illustratif

En considérant l'exemple illustratif, on peut avoir [*Histoire* : *1H202*, *plutôtVrai*] et [*Géographie* : *1H202*, *trèsFaux*]. Les modalités *plutôtVrai* et *trèsFaux* sont définies au niveau ontologique et ne sont donc pas spécifiques à un nœud individuel. Elles sont associées à un sous-ensemble flou dans K et plus particulièrement dans T et F respectivement.

Cet exemple peut représenter *Il est* plutôt vrai *que le cours* 1H202 *traite d'*histoire *et il est* très faux *que* 1H202 *traite de* géographie.

2.2.2.3 Interprétation

Plusieurs remarques sont à discuter autour de ce modèle, et permettent à la fois de donner un nouvel éclairage à ce modèle et d'en montrer quelques limites.

Une motivation erronée ? Comme évoqué plus haut, Wuwongse et Cao, 1996 motivent leur utilisation de valeurs linguistiques au lieu de valeurs numériques en argumentant

que, dans le cas contraire des résultats *non raisonnables* peuvent être obtenus.

Ce qui suit est l'exemple donné par l'article. Soient deux concepts $A = [t_a, a|c_a]$, $B = [t_b, b|c_b]$ où $(a,b) \in I^2$ $(t_a, t_b) \in T_C^2, t_a \leq t_b$, $(c_a, c_b) \in K^2, c_a \leq c_b$. Alors A est une projection de B et donc A implique B d'après les définitions de la projection telles que fournies par Morton, 1987 d'après Wuwongse et Manzano, 1993 par exemple. Wuwongse et Cao, 1996 considèrent que ce n'est *pas raisonnable* en l'illustrant avec l'exemple de [*Garçon* : *Cyndy*|0] qui ne devrait pas impliquer [*Garçon* : *Cyndy*|1]. Le diagnostic de Wuwongse et Cao, 1996 est que cela est dû à la compatibilité (la valeur de vérité) comme étant définie comme un réel dans $[0,1]$, et ils proposent de résoudre ce problème par l'utilisation d'un sous-ensemble flou sur $[0,1]$.

Le choix d'ajouter des variables linguistiques au lieu de valeurs numériques est motivé par ce problème de raisonnement, cependant il semble que l'on peut a contrario garder les valeurs réelles dans $[0,1]$ pour la pondération d'un nœud concept flou, et uniquement changer une condition dans la relation de projection, comme proposé par l'article. Ainsi, au lieu d'exiger que c_a soit plus petite que c_b afin que A implique B, on peut imposer que c_b soit plus petite que c_a. On obtiendrait alors [*Garçon* : *Cyndy*|1] qui implique [*Garçon* : *Cyndy*|0].

Ce sont plutôt trois choix effectués par l'article qui contournent le problème. D'une part les différentes valeurs sont typées en fonction de leur appartenance aux classes T, F ou U. D'autre part, ces valeurs sont prises en compte dans la relation de projection proposée. Enfin, un degré de non-correspondance (*mismatching degree*) est défini, similaire par exemple au degré de certitude (Kowalski & Martin, 2019) du cadre incertain. La relation d'ordre entre valeurs floues u et v sur l'univers U, de fonctions d'appartenance respectives μ_u et μ_v, devient associée à une valeur de non-correspondance :

$$\sup_{x \in U} \{\max\{\mu_u(x) - \mu_v(x), 0\}\}$$

Ces trois éléments sont pris en compte dans la relation de projection et il n'est donc pas exact de dire que l'utilisation seule de sous-ensembles flous au lieu de valeurs réelles suffit.

Modificateurs linguistiques (3a)

Définition En premier lieu, K et les trois ensembles T, F et U qui le constituent peuvent être formulés comme trois modalités linguistiques, *Vrai*, *Faux* et *Inconnu*, associées à des modificateurs linguistiques (Bouchon-Meunier, 1992). En effet, les éléments d'une même classe, disons T par exemple, partagent une valuation commune, dans l'exemple une valuation positive, et ne diffèrent pas dans les exemples partagés

dans l'article Wuwongse et Cao, 1996 que par le modificateur qui leur est associé. Le cas de U est particulier et n'est pas compris dans cette interprétation.

Exemple En reprenant l'exemple du cours d'histoire, [*Histoire* : *1H202*, *plutôtVrai*] devient [*Histoire* : *1H202*, (*plutôt*, *Vrai*)]

Interprétation Dans l'ensemble T on a ainsi les éléments :

$$\{trsPeuVrai, peuVrai, plutôtVrai, Vrai, trèsVrai, \ldots\}$$

où on peut distinguer alors $\{pas, peu, plutôt, très, \ldots\}$ comme un ensemble de modificateurs linguistiques et la modalité linguistique *Vrai*. (3*a*)

Ainsi, cette interprétation peut impliquer que le modèle de graphe conceptuel flou proposé par Wuwongse et Cao, 1996 n'admet que trois modalités linguistiques, *Vrai*, *Faux* et *Inconnu*, correspondant respectivement aux ensembles T, F et U. Cette interprétation peut se représenter par une partition de K en $\{pas, peu, plutôt, très\}$.

Contrainte relâchée En suivant cette interprétation (3*a*), les nœuds concept flous peuvent être assimilés à des variables linguistiques à modalités dans K. La contrainte relâchée correspond à la valeur de vérité d'un nœud, qui était à valeur dans $\{0, 1\}$ (il est ou il n'est pas) et devient à valeurs dans $K = \{V, F, U\}$ selon trois modalités linguistiques, et un certain nombre de modificateurs pour V et F.

Monde ouvert, monde clos (3b) Les sous-ensembles flous associés aux modalités linguistiques λ sont des éléments de l'ensemble K définis au niveau du vocabulaire. Sous l'hypothèse de monde ouvert, qui spécifie que tout ce qui n'est pas défini peut être vrai, alors la définition peu contrainte de K permet de définir une infinité de modalités.

Ceci tempère l'affirmation précédente concernant l'utilisation plus intuitive des modalités linguistiques. Cependant, en suivant l'interprétation (3*a*), qui stipule que seules 3 modalités sont possibles, on peut supposer qu'il existe une infinité de modificateurs linguistiques (3*b*1).

Sous l'hypothèse de monde fermé, qui spécifie que tout ce qui n'est pas défini est faux, alors il y a besoin de préciser les modalités possibles dans la définition de K (3*b*2).

Marqueur générique (3c) Enfin, la possibilité d'utiliser un marqueur générique, contrairement au modèle proposé par Morton, 1987, suppose que l'imprécision (ou l'incertitude) représentée n'est pas inhérente aux couples dans $T_C \times I$. En effet, cette possibilité peut impliquer que la modalité linguistique associée à un nœud générique, [*Histoire* : $*$, *vrai*] par exemple, devient définie comme uniquement fonction du type.(3*c*1)

Cette interprétation dépend de la sémantique du marqueur générique comme une entité non définie correspondant à un quantificateur universel. Une interprétation en tant que quantificateur existentiel transpose l'inhérence de l'imprécision (ou l'incertitude) aux couples dans $T_C \times \mathcal{M}$ ($3c2$).

Ainsi, Wuwongse et Tru relâchent également la contrainte sur la précision (ou la certitude) de l'entité représentée (elle peut être moins précise grâce à l'utilisation du marqueur générique) et sur la précision des valeurs de compatibilité (elles sont en effet issues de la logique floue de type-II).

2.2.3 Pondération ontologique

Un troisième type de pondération des nœuds concept utilise des poids définis dans le vocabulaire. On peut noter que l'interprétation ($3b$) ancre la précédente définition dans le même cas. Cependant, la présente section ne laisse pas d'ambiguïté à cette interprétation : les poids sont directement définis dans le vocabulaire, et la partie factuelle redevient des graphes conceptuels classiques.

2.2.3.1 Syntaxe

Le modèle, proposé par Cao et al., 1997, ne représente pas les types flous au niveau de la connaissance factuelle, mais définit d'une part un treillis de types basiques et d'autre part un treillis de valeurs de vérité linguistiques. En prenant c dans le treillis de types basiques (c'est-à-dire non flous) défini T_C, et λ dans le treillis de valeurs de vérité K, on obtient pour $m \in \mathcal{M}$:

$$[c_\lambda : m], \lambda \in K \tag{2.3}$$

Un élément à prendre en compte, également présent dans la précédente proposition par Wuwongse et Cao, 1996, est l'utilisation d'un treillis de types plutôt qu'un ensemble ordonné, comme le propose Sowa, 2008, ce qui tranche avec le choix effectué dans le formalisme des graphes conceptuels de Chein et Mugnier, 2008, et modifie notamment l'interprétation d'intersection de type.

En effet, Chein et Mugnier, 2008 et Cao, 2010 explicitent la distinction de treillis de types selon que l'interprétation de ce treillis est basée sur la théorie des treillis ou la théorie des ordres sur les ensembles. Référençant le travail de Beierle et al., 1992, sachant qu'un type est interprété comme l'ensemble des objets qui lui appartiennent, la distinction est effectuée au niveau de l'interprétation de l'intersection de deux types, c'est-à-dire le type commun le moins spécifique : dans le cas de la théorie des treillis, une telle intersection est considérée comme l'intersection des interprétations de chacun des deux types ; dans le cas de la théorie des ordres, l'intersection est interprétée comme

un sous-ensemble des interprétations. La précédente interprétation est préférée dans les cas pratiques. En effet, par exemple, si "Attaque web" est le sous-type de "Attaque client" et "Attaque serveur" le moins spécifique, il ne recouvre pas nécessairement toutes les attaques qui appartiennent à la fois à "Attaque client" et "Attaque serveur", c'est-à-dire qu'il peut exister en pratique une attaque qui fait partie de ces deux types sans appartenir au type "Attaque Web".

Par ailleurs, la différence entre les définitions de nœud concept flou par Wuwongse et Cao, 1996 et Cao et al., 1997 réside dans une relation entre type et modalité ontologique, et sa définition est alors antérieure à la relation entre type et marqueur.

2.2.3.2 Exemple illustratif

Dans le cas de l'exemple considéré, on peut par exemple construire le nœud concept $[Histoire_{trèsVrai} : 1H202]$.

Si $Histoire_{trèsVrai}$ est le plus grand sous-type du type $CoursMagistral_{trèsVrai}$ et du type $Histoire_{Vrai}$, il recouvre l'ensemble des instances avec la première définition utilisant un treillis et un sous-ensemble avec la seconde définition utilisant un ensemble. Dans ce dernier cas, il peut donc exister des cours qui sont majoritairement des cours magistraux traitant plutôt d'histoire, mais dont on sait qu'ils ne traitement pas majoritairement d'histoire, et cela n'est pas possible dans le premier cas.

2.2.3.3 Interprétation

Cao, 2010 adapte la définition de pondération ontologique en remplaçant le treillis de types basiques par un ensemble de types basiques disposant d'une relation d'ordre.

Un graphe conceptuel flou utilisant de tels types de concept flous est très similaire à un graphe conceptuel classique. En effet, les nœuds concept sont des couples (type non flou, marqueur) dont le type appartient à un ensemble avec une relation d'ordre partiel. La modalité linguistique λ de l'équation 2.3 n'est pas déterminée lors de la construction du nœud factuel, mais dans le vocabulaire, et elle est associée à un type.

Ainsi différentes occurrences du type de base *Histoire* peuvent être associées à différentes valeurs de λ, résultant en différents types flous.

Par cette proposition, comme pour (2*b*), la pondération est explicitement liée aux types, et l'imprécision est située au niveau de la connaissance ontologique (4).

De manière similaire aux interprétations (3*a*), (3*b*) et (3*c*), on peut préciser les interprétations (4*a*), (4*b*) et (5*c*) résultantes.

2.3 Valeur d'attribut imprécise

Cette section traite d'un autre cas d'intégration sémantique floue : le flou est défini à l'intérieur d'un nœud concept, mais contrairement aux cas précédents, pour un type particulier de concept, appelé concept avec attribut dans le cadre des graphes conceptuels classiques. Ces nœuds concept particuliers contiennent une valeur d'attribut en plus, ou à la place, d'un marqueur. L'extension floue de tels nœuds permet de définir des modalités linguistiques en tant que valeurs d'attribut : ces dernières deviennent alors des variables linguistiques.

Après avoir rappelé la définition de ces types de concept avec attribut dans le cas non flou, les sections 2.3.2, page 58 et 2.3.3, page 60 discutent successivement plusieurs extensions floues proposées dans la littérature. La seconde a comme particularité d'être formulée comme la définition de marqueur flou, mais elle représente des valeurs floues et le choix de les modéliser en marqueur est dans un souci d'homogénéité des formalisations. Le tableau 2.3 propose une synthèse des trois extensions considérées dans cette section.

2.3.1 Rappel : type de concept avec attribut

Dans le formalisme classique des graphes conceptuels présenté chapitre 1, les types de concept avec attribut (Chein & Mugnier, 2008) constituent un type de concept associé à un domaine de valeurs. Un nœud concept avec attribut est noté $[c : v]$ avec c le type de concept avec attribut et v une valeur.

Selon le cas, un sous-ensemble T_C^a de T_C peut être défini pour regrouper ces types de concept avec attribut, et le choix peut être fait ou non d'inclure le domaine de valeurs dans l'ensemble I des marqueurs individuels. Toutefois dans le cas de l'inclusion de valeurs numériques réelles par exemple, I pourrait alors devenir infini. Dans tous les cas, tout type de concept avec attribut $c \in T_C$ est associé à son domaine de valeur noté $U(c)$ d'un univers de discours U dans la suite de ce manuscrit.

Par exemple, on peut représenter $[Note : 18]$, où *Note* est un type de concept avec un domaine de valeurs $U(Note) = [0, 20]$ et 18 est une valeur dans $U(Note)$.

Dans le formalisme de Sowa, 1983, les valeurs sont des symboles, et les marqueurs des nœuds concept avec attribut sont également présents : tout nœud concept a un champ marqueur et un champ valeur. Par défaut le marqueur est le marqueur générique $*$ et la valeur est 1.

Buche et al., 2001 introduisent une définition alternative avec l'ajout d'un type de concept et d'un type de relation particuliers, respectivement *Value* et *Val*. Un nœud relation de type *Val* relie un nœud concept classique à un nœud concept de type *Value* ayant

TABLE 2.3 – Valeur ou marqueur pondéré

Modèle	Définition	Exemple
Valeur métrique floue (MORTON, 1987) 2.3.2, page 58	Représentation de l'imprécision du langage naturel Valeur $v \subseteq_f D_v$ où D_v est le domaine de valeurs sur l'univers U Attribut métrique, c'est-à-dire ayant une mesure associée	$[Note : Bonne]$ $\exists\ \mu_{Bonne} : U(Note) \longrightarrow [0,1]$ peut s'interpréter comme une bonne note, et correspond aux notes non floues avec les degrés d'appartenance donnés par μ_{Bonne}.
Valeur non-métrique floue (WUWONGSE & MANZANO, 1993) 2.3.2, page 58	Extension du modèle de valeur métrique floue au cas d'une valeur non-métrique λ 2 ajouts : * U remplacé par un univers discret U' * $\lambda \subseteq_f U'$ de fonction caractéristique $\mu_\lambda : U' \longrightarrow [0,1]$	$[Couleur : Rouge]$ peut représenter la couleur *Rouge*, en tant que sous ensemble flou d'un univers de discours discret U', correspondant aux différentes longueurs d'ondes du spectre visible.
Définition homogène de marqueur flou (THOMOPOULOS et al., 2003b) 2.3.3, page 60	Variante des deux modèles précédents 4 différences : * plus d'univers U ou U' : valeur $v \in I$ * valeur floue $v \subseteq_f I, \forall\ m, \mu_v(m) > 0 \implies \tau(v) \geq \tau(m)$ * cas métrique et non-métrique confondus * valeur normalisée	$[Note : Bonne]$ peut représenter une bonne note *Bonne* n'a pas de mesure μ_{Bonne} associée, c'est une disjonction de marqueurs individuels de type *Note* pondérés par une valeur dans $[0,1]$.

une valeur à la place du marqueur individuel : une note u par exemple est représentée comme :

$$[Note : maNote] - 1 - (Val) - 0 - [Value : 18]$$

Nous proposons d'utiliser une notation simplifiée, telle que pour l'exemple $[Note : 18]$, et telle qu'utilisée dans Thomopoulos et al., 2003b.

2.3.2 Modalité linguistique en tant que valeur

Les nœuds concepts avec attributs ont été étendus pour permettre la représentation de valeurs floues, d'abord sous la forme de modalité linguistique.

Syntaxe

Morton, 1987, d'après Wuwongse et Manzano, 1993, permet la représentation de connaissances imparfaites dans le champ valeur d'un nœud concept avec attribut.

$$[c : \lambda], \lambda \in \mathcal{F}(U(c)) \tag{2.4}$$

où $\mathcal{F}(U)$ désigne l'ensemble des sous-ensemble flou définis sur le domaine U. La connaissance imparfaite est ainsi représentée par une modalité linguistique λ associée à un sous-ensemble flou sur le domaine de valeurs $U(c)$.

Morton, 1987, d'après Wuwongse et Manzano, 1993, effectue une distinction entre attributs métriques et attributs non-métriques. Un attribut métrique serait un attribut auquel une mesure est associée. Ainsi on peut distinguer deux cas :

Dans le premier cas, considérons la mesure *Taille* à valeurs dans $U(Taille)$. Alors, on aurait par exemple *Grand* qui instancie *Taille* par la fonction d'appartenance μ_{Grand} : $U(Taille) \longrightarrow [0,1]$.

Dans le second cas, il n'y a pas de mesure associée, et une valeur *Rouge* est un terme flou auquel on associe un sous-ensemble flou d'un univers discret U' ayant pour fonction d'appartenance $\mu_{Rouge} : U' \longrightarrow [0,1]$.

Cependant, la distinction reste obscure, étant uniquement illustrée par un exemple : elle pourrait correspondre à une distinction entre valeurs numériques (avec mesures) et valeurs catégorielles (sans mesure), mais on ne peut que l'induire de l'exemple.

Exemple illustratif

Par exemple, si *Note* est métrique, on peut construire le nœud $[Note : Bonne]$ où *Bonne* est une modalité linguistique associée à un sous-ensemble flou de l'ensemble $[0,20]$. La fonction d'appartenance de *Bonne* est déterminée par la mesure associée à *Note*. On peut par exemple imaginer une mesure *Notation* qui dans ce cas est instanciée par la modalité *Bonne* par la fonction d'appartenance $\mu_{Bonne} : U(Notation) = [0,20] \longrightarrow [0,1]$.

Interprétation

La distinction entre les cas métriques et non-métriques semble à la fois précise et ambiguë. La définition est précise tant elle spécifie qu'il suffit tout simplement qu'une mesure soit définie. La définition est cependant ambiguë étant donné que seul un exemple et quelques éléments permettent de caractériser le cas non-métrique (5).

Valeur Dans l'article original de Wuwongse et Manzano, 1993 retranscrivant Morton, 1987, il faut se munir dans le cas métrique d'une fonction pour chaque nœud concept avec attribut métrique $c = [t : m, \mu_c]$, comme pour le cas de la fonction de compatibilité d'un nœud concept flou présenté dans le même article. La fonction associée correspond à la mesure associée au type t, a pour ensemble de définition $\mathcal{M}$ et est à valeurs dans $U(t) \cup \ U^{\$}(t) \cup \{?\}$ où ? est la valeur non définie et $U^{\$}(t)$ est l'ensemble des sous-ensemble flou de $U(t)$.

La contrainte relâchée ici est la possibilité d'utiliser des valeurs floues en tant que valeur.

Cette définition, comme pour l'interprétation (2*a*), participe d'une définition locale des connaissances représentées dans les graphes conceptuels. (5*a*)

Définir une fonction d'appartenance μ_c spécifique à chaque nœud concept avec attribut métrique semble contredire l'idée qu'un même nœud comprend une seule valeur. Il est pourtant plutôt explicite, tant dans la définition en langage naturel que la définition en notation formelle, qu'une seule valeur est associée à tout élément de M au sein d'un nœud donné d'une part, et que d'autre part chacun de ces nœuds ne dispose que d'un seul marqueur dans M. Pour clarifier, il aurait fallu soit définir cette fonction globalement comme pour le cas (2*b*), soit directement associer une valeur plutôt qu'une fonction à chaque nœud. (5*b*)

Emplacement Les modèles définis par Morton, 1987 et de Wuwongse et Manzano, 1993 ne permettent pas de se prononcer aisément sur la sorte de connaissance impactée par leur extension floue : factuelle ou ontologique. D'une part, il existe la possibilité que le domaine de valeurs $U(c)$ d'un type attribut t renvoie un sous-ensemble flou sur U, ce qui est conforme à une inclusion sur la partie ontologique si les mesures associées sont définies dans le vocabulaire.

Cependant, cette possibilité n'est pas explicitée avant son utilisation dans un graphe conceptuel, car aucun élément supplémentaire n'est défini dans la partie ontologique dans l'article. En effet, il n'est pas explicite que les modalités linguistiques définies sur $U(c)$ soient définies avant leur utilisation dans la partie factuelle ou non.

2.3.3 Marqueur flou

Une définition similaire à la précédente, proposée par Thomopoulos et al., 2003b, introduit la notion de *Marqueur flou*. Elle est basée sur le modèle de valeur dans un graphe conceptuel proposé par Buche et al., 2001. Elle diffère du modèle de Buche et al., 2001 et de la proposition de Morton, 1987 en ce que le domaine de tout type de concept attribut est défini sur I, l'ensemble des marqueurs individuels, et non un univers U distinct de I.

Syntaxe

Ainsi, un marqueur flou est un sous-ensemble flou sur I restreint au domaine du type de concept associé t_c dans le nœud concept c. I comprend ainsi en plus des marqueurs individuels classiques des marqueurs correspondant à des valeurs, telles que des valeurs numériques. Un marqueur classique est le cas particulier d'un ensemble non flou qui associe 1 comme degré d'appartenance à l'élément considéré m de I sur le domaine de t_c, et 0 aux autres.

Les valeurs sont des éléments de I nécessairement associés au type introduit par l'article noté *Valeur*, ou un de ses sous-types. Les valeurs numériques sont nécessairement associées au second type introduit *ValeurNumérique*, sous-type de *Valeur*, ou un de ses sous-types. Une fonction, $Ref : T_C \longrightarrow I$ associe à chaque type t l'ensemble des marqueurs individuels qui lui sont conformes, c'est-à-dire que le type maximal qui leur est associé est t ou une de ses spécialisations. Un marqueur flou associé au type t est ainsi ici un sous-ensemble flou sur $Ref(t)$.

Ce modèle permet de représenter l'imprécision en passant d'une seule valeur précise à un ensemble de valeurs pondérées. Dans ce cas une fonction de conformité inverse à celle classiquement définie dans les graphes conceptuels est utilisée. Au lieu de définir une fonction τ associant un type dans T_C à tout élément de I, Thomopoulos et al., 2003b utilisent une fonction $Ref(t) : I \longrightarrow \{0,1\}$ renvoyant les marqueurs conformes à chaque type, et la généralisent dans le cas flou en renvoyant un sous-ensemble flou sur $[0,1]$. Ces deux fonctions semblent néanmoins équivalentes en terme d'interprétation. Il y a une homogénéité entre la définition des marqueurs compatibles avec un type donné, et le domaine de valeur associé à un type de concept attribut.(5*c*)

Exemple illustratif

Par exemple, dans le cas du nœud $[Note : maNote]-1-(Val)-0-[Value : 18]$ représenté à la manière de Buche et al., 2001, on fuzzifie sa valeur selon ce modèle en remplaçant 18 par un sous-ensemble flou sur $Ref(Note)$.

Interprétation

On peut considérer que la définition de Thomopoulos et al., 2003b conduit à une confusion entre le marqueur individuel i, qui fait référence à une entité dans la réalité, et une valeur v, qui correspond à la mesure d'une caractéristique. D'un point de vue syntaxique, ce modèle ne fait pas de distinction entre les types de concept avec attribut et les types de concept classiques, si ce n'est par la définition du type *Valeur* comme type de concept avec attribut le plus général. Cependant, leur sémantique reste bien différente, car ces deux types de concept ont une interprétation différente dans la réalité. En effet, on distingue entre la valeur d'un attribut et le marqueur symbolique faisant référence à une entité dans la réalité.

Un élément peut modérer cette ambiguïté : dans le contexte d'application présenté par Thomopoulos et al., 2003b, où les graphes conceptuels sont utilisés pour représenter des données en microbiologie, la confusion peut ne jamais se produire en pratique, mais ce n'est pas dit explicitement.

Une première modification de I est le fait que ses éléments ne font plus nécessairement référence à des entités, mais à la fois à des entités et à des valeurs d'attribut. Autrement dit, les valeurs d'attribut sont alors considérées comme des entités.

Une seconde modification est qu'il est alors possible de définir des domaines de valeurs sur I, discrètes ou non, continues ou discontinues.

Cette définition est sans ambiguïté sur la partie ontologique en ce qui concerne les domaines de valeurs, et sur la partie factuelle en ce qui concerne les sous-ensembles flous sur ces domaines de valeurs, puisqu'ils ne sont associés à aucune variable ou élément du vocabulaire.

En conclusion, deux contraintes sont relâchées : les valeurs peuvent désormais être des sous-ensemble flous sur I, les éléments de I ne sont, par voie de conséquence, plus nécessairement des références à des individus.

2.4 Nœud multi-concept flou

Un troisième type d'intégration de connaissances imparfaites est défini dans le cas des nœuds multi-concept, qui correspondent classiquement à des nœuds conjonctifs, rappelés dans la section 2.4.1. Trois variantes floues sont ensuite présentées successivement en sections 2.4.1, 2.4.2 et 2.4.3. Une synthèse en est offerte dans le tableau 2.4.

TABLE 2.4 – Conjonction ou disjonction pondérée

Modèle	Définition	Exemple
Conjonction floue de types (CAO, 1999) 2.4.1, page 63	Utilisation conjointe des extensions du nœud concept flou avec pondération globale et des types conjonctifs Chaque type est donc par défaut une conjonction d'éléments incomparables de T_C^f	$[Histoire_{vrai}, Géographie_{peuVrai} : 1H002]$ où $vrai \in T$, $peuVrai \in T$ $vrai \geq peuVrai$ peut représenter le cours *1H002* qui traite d'*histoire*, et traite un peu de *géographie*, via l'étude de situations historiques par des cartes par exemple.
Définition homogène de disjonction floue de types (THOMOPOULOS et al., 2003b) 2.4, page 61	Disjonction de types incomparables de T_C Chaque type au sein de la disjonction est pondéré par une valeur dans $[0,1]$ Marqueur d'un nœud concept comportant une telle disjonction nécessairement générique $*$	$[(Histoire, 0.8), (Géographie, 0.4) : *)]$ peut symboliser *L'ensemble des cours dont 80% du contenu traite d'*Histoire *et 40% traite de* Géographie. Une interprétation floue est que c'est une contrainte du cours qu'il y ait au moins 80% d'histoire ou au moins 40% de géographie
Hiérarchie floue (THOMOPOULOS et al., 2003a) 2.4.3, page 65	Extension du cas de la disjonction floue de types Différence : propagation des pondérations pour les éléments de T_C extérieurs à la disjonction déduits des poids de la disjonction	$T_C = [(Cours, 0), (Histoire, 0.8), (Géographie, 0.4), (Histoire - Géographie, \alpha)]$ hiérarchie construite à partir de l'exemple précédent où les relations d'ordre sont explicites et les poids propagés. *Cours* prend la valeur 0 en tant que type plus général que chaque élément de la disjonction. *Histoire – Géographie*, le plus spécialisé, a $\alpha = 0.8$ ou $\alpha = 0.4$ selon une interprétation en préférence ou précision.

2.4.1 Conjonction floue de types

Rappel du cas non flou

Les nœuds multi-concept étendent la définition classique des graphes conceptuels en permettant l'utilisation de types conjonctifs (Cao, 1999 ; Baget, 2003a ; Chein & Mugnier, 2008). La fonction d'étiquetage *label* des nœuds associe les nœuds concept à un sous-ensemble de T_C de types incomparables au lieu d'un type unique. Deux types sont dits incomparables s'ils ne peuvent pas être comparés par la relation d'ordre partiel définie sur T_C, c'est-à-dire que l'un n'est pas la généralisation de l'autre et vice versa.

Ainsi, en reprenant l'exemple des attaques informatiques, on peut considérer le type conjonctif (*AttaqueDangereuse*, *AttaqueRapide*) qui recoupe les attaques informatiques qui sont à la fois dangereuses et qui se sont déroulées en un temps réduit.

Il faut noter que, sous certaines hypothèses, le type conjonctif ne correspond pas à un accroissement de l'expressivité mais à un raccourci syntaxique dans la partie factuelle des graphes conceptuels : un nœud avec un type conjonctif c constitué de n types incomparables peut être remplacé par n nœuds dont les types respectifs sont les n types constituant le type conjonctif c.

Ainsi, le nœud [(*AttaqueDangereuse*, *AttaqueRapide*) : *Attaque12*] est le raccourci syntaxique des nœuds [*AttaqueRapide* : *Attaque12*] et [*AttaqueDangereuse* : *Attaque12*].

Cas flou

Syntaxe. Cao, 1999 propose une extension naturelle au cas flou, en remplaçant les concepts considérés par des concepts flous tels que présentés section 2.2.3, page 54. Cette extension définit une conjonction de tels concepts flous, et on obtient avec c et c' dans T_C, λ et λ' dans K, m dans M :

$$[c_\lambda, c'_{\lambda'} : m] \tag{2.5}$$

De la même manière que pour le type conjonctif, il s'agit d'un raccourci syntaxique qui n'augmente pas l'expressivité par rapport aux propositions sur les nœuds et les types flous.

Dans le cas classique, il est naturel d'utiliser la conjonction dans le cas où définir un type la recouvrant ayant un sens intuitif n'est pas toujours possible. De plus, la conjonction permet de définir une multiplicité de combinaisons de types sans explicitement les définir préalablement dans le vocabulaire, via un sous-type commun.

Dans le cas flou, cela est d'autant plus essentiel sur le modèle de treillis de type flous (Cao et al., 1997) (ou d'ensemble partiellement ordonné de types flous (Cao, 2010)) car il y a d'autant plus de cas à considérer avec toutes les combinaisons possibles de types non flous et valeurs.

Exemple illustratif. Par exemple, on peut construire :

$$[Histoire_{vrai}, Géographie_{peuVrai} : 1H002]$$

où *1H002* peut être une leçon d'histoire-géographie ne présentant que quelques cartes pour illustrer la partie histoire.

Interprétation. Il est à noter que dans cet exemple la conjonction de types flous utilisée est pertinente, elle a un sens conceptuel, mais ce n'est pas toujours le cas. Ainsi, la conjonction de *Étudiante* et *Cours* n'est pas conceptuellement claire. Cependant, cette conjonction peut se produire et être pertinente dans la pratique. Par exemple, il pourrait arriver, en études d'art disons, qu'un étudiant suive à la fois certains cours et soit d'autre part l'objet d'un cours en particulier. Il serait ainsi dans plusieurs occurrences une instance de *Étudiante*, et dans une occurrence une instance de *Cours*. En définitive, la conjonction de types flous a parfois un sens, même inattendu, et dépend de l'exploitation faite des connaissances, mais en règle générale les conjonctions floues qui ont du sens sont une minorité.

La contrainte relâchée dans ce cas est la capacité à représenter des nœuds comportant une conjonction pondérée de types plutôt qu'un type unique non-pondéré. Cependant, en comparant aux cas (2*c*) à l'extension classique rappelée en section 2.4.1, la définition ici ne relâche pas plus de contraintes. (6)

2.4.2 Disjonction floue de types

Syntaxe

Dans un cas différent de multi-concept flou, Thomopoulos et al., 2003b proposent de définir la disjonction floue de types incomparables, en donnant pour c et c' deux types, et α et β deux valeurs dans $[0,1]$:

$$[(c,\alpha),(c',\beta) : *] \tag{2.6}$$

Le contexte de cette proposition est l'utilisation d'un graphe conceptuel comme requête dans des bases de données, où les résultats doivent correspondre au graphe conceptuel de la requête, c'est-à-dire être égaux à celui-ci ou en être une spécialisation. La disjonction définie correspond alors à une imprécision ou à une préférence sur la nature de l'information recherchée, et permet ainsi des requêtes plus expressives renvoyant un plus large éventail de résultats.

Les résultats doivent correspondre au type disjonctif, c'est-à-dire être du même type ou d'un sous-type, proportionnellement à l'importance du poids associé à chaque type.

Le choix de l'interprétation en tant qu'imprécision ou en tant que préférence modifie la notion de distance avec le graphe conceptuel de la requête, et renvoie donc des résultats différents. Dans ce paradigme, la pondération peut se voir comme un marqueur générique associé à un sous-ensemble flou de I. Ce sous-ensemble flou représente tous les marqueurs individuels correspondant à la disjonction floue des types, c'est-à-dire subsumés par un tel sous-ensemble flou, et où chaque marqueur a un degré d'appartenance égal au degré du type correspondant dans la disjonction floue.

Exemple illustratif

Suivant la définition de Thomopoulos et al., 2003b, on peut par exemple construire le nœud $[(\textit{Histoire}, 0.8),\ (\textit{Géographie}, 0.4) : *)]$ qui peut représenter les leçons qu'une étudiante préférerait avoir l'année suivante.

Interprétation

Par rapport aux définitions conjonctives précédentes, il faut souligner que la disjonction ne se réduit pas à un raccourci syntaxique. En effet, l'équivalence entre un nœud multi-concept dans le cas conjonctif et plusieurs nœuds n'est pas possible dans le cas disjonctif. Il n'existe à notre connaissance aucune autre représentation de la disjonction de connaissance dans le formalisme des graphes conceptuels permettant une telle reformulation syntaxique.

La contrainte relâchée par cette définition est le degré de vérité associé à chaque type. De plus, un niveau supplémentaire de flou est ajoutée avec le relâchement de la contrainte sur l'unicité du type. En effet, alors que dans les cas de types mono-concept, un nœud est associé à un type unique, ici, on ne peut pas établir que l'entité représentée est de tel ou tel type. Un ensemble flou sur un sous-ensemble de types incomparables de T_C est alors déterminé au lieu d'un seul élément de T_C. (*7a*)

2.4.3 Hiérarchie floue

Syntaxe

Thomopoulos et al., 2003a étendent la définition précédente. Depuis la disjonction floue de type présente dans la requête, des poids sont déduits pour tous les types de T_C, c'est-à-dire même ceux qui ne sont pas présents dans la disjonction. Il s'agit d'une forme dite *développée* nommée *type flou en extension* dans l'article qui la propose, par opposition à la disjonction floue de types, correspondant à la définition précédente, nommée *type flou en intention*.

Selon que l'interprétation soit la préférence ou l'imprécision, on choisit récursivement le maximum ou le minimum des supertypes pour chaque sous-type, et 0 pour les types plus généraux, comme l'illustre l'exemple ci-après. L'idée est qu'une requête sous forme de graphes conceptuels flous spécifiant une préférence caractérise une requête permissive, et donc le maximum est attribué, alors qu'une requête spécifiant une précision caractérise une requête restrictive, et donc le minimum est attribué.

Il s'agit d'une définition de type flou qui est spécifique au cas d'interrogation d'une base de données de graphes conceptuels présentée dans la section 2.3.3, page 60.

D'autres propositions (Thomopoulos et al., 2006) imposent d'être plus permissif en prenant systématiquement le maximum, afin de ne perdre aucun résultat éventuellement pertinent, quelle que soit l'interprétation choisie.

Exemple illustratif

Ainsi, en considérant une disjonction floue de types au sein d'un nœud concept tel que $[(Histoire,\ 0.8), (Géographie,\ 0.4) :\ *)]$, on peut construire la hiérarchie :

$$T_C = [(Histoire,\ 0.8), (Géographie,\ 0.4), (Cours,\ 0), (Histoire - Géographie, \alpha)]$$

où le type *Cours* est plus général que les types *Histoire* et *Géographie*, tandis que le type d'étiquette *Histoire – Géographie* est plus spécifique que les types *Histoire* et *Géographie*. Le coefficient α prend la valeur 0.8 dans le cas d'une représentation de préférence, et 0.4 dans le cas d'une représentation d'imprécision.

Interprétation

La manière de construire une hiérarchie floue dépend par conséquent du type de connaissance imparfaite représentée. Contrairement à toutes les définitions précédentes, dans ce modèle proposé par Thomopoulos et al., 2003a, une distinction est effectuée selon que les connaissances représentées caractérisent une préférence ou une imprécision. En effet, les définitions de graphe conceptuel flou précédentes proposent des interprétations possibles, et ne traitent l'influence de ces interprétations que dans l'étape d'inférence du raisonnement. Elle rend explicite l'interprétation de ce que représente une disjonction floue de type sur T_C. Au delà de ça, aucune contrainte supplémentaire n'est relâchée par rapport au cas (7*a*) précédent. En effet, en un sens, elle ne fait que déduire de la disjonction des connaissances sur les types non présents dans la disjonction selon un mécanisme de raisonnement qui lui est propre. (7*b*)

TABLE 2.5 – Autres modèles de flou

Relation conceptuelle floue (WUWONGSE & MANZANO, 1993) 2.5.1, page 68	Représente la compatibilité des nœuds concept connexes avec la relation Degré de compatibilité dans $[0,1]$ pour tout $r \in R$	(*Construit*, 0.2) -0-[*Humain* : *Moi*] -1-[*Maison* : *MaMaison*] signifie *Il n'est pas totalement inexact de dire que j'ai construis ma maison.*
Flou propositionnel (MORTON, 1987) 2.5.2, page 69	Représentation de la compatibilité à la description d'un nœud concept Fonction de compatibilité : $\forall\ v \in\ V, \mu_v$: $[0,1] \longrightarrow [0,1]$	[Maison : (MaMaison : (Descr, 0.8))] représente un nœud concept contenant une description précise, sous forme de graphe conceptuel, de ma maison.
Règle floue (WUWONGSE & CAO, 1996 ; CAO & CREASY, 2000) 2.5.3, page 70	Règles-λ dont l'hypothèse et la conclusion sont des graphes conceptuels-λ flous Dans CAO et CREASY, 2000 la règle-λ remplace un graphe conceptuel comprenant un quantificateur universel. Dans le cas flou, la sémantique du quantificateur devient une variable linguistique tel que *la plupart*	(*assister*) -0-[*Étudiant* : *la plupart*] -1-[*Histoire* : *1H003*] **devient :** *Étudiant* : $*x$ $\Longrightarrow$ *laPlupart* (assister) -0-[*Étudiant* : $*x$] -1-[*Histoire* : *1H003*]

2.5 Autres fuzzifications

Cette section traite d'autres cas d'inclusion de connaissances imparfaites qui ne donnent pas lieu à autant d'analyse que pour les autres définitions, soit en raison de leur simplicité, soit parce qu'ils ne sont qu'une transposition de cas détaillés précédemment. Nous abordons d'abord le cas des relations floues, puis celui du flou propositionnel et enfin le cas des règles d'inférence floues. Elle sont résumées dans le tableau 2.5.

2.5.1 Relations floues

L'intégration de connaissances floues peut s'appliquer aux types de relation, comme pour les types de concept discutés dans la section 2.2, page 41, par transposition directe et menant aux même discussions. En effet, les poids associés aux concepts décrits dans la section 2.2, page 41, qu'ils soient numériques ou linguistiques, peuvent être naturellement étendus aux relations.

Poids en tant que valeur numérique

Wuwongse et Manzano, 1993 proposent de généraliser le principe discuté dans la section 2.2.1, page 42 pour le cas des nœuds concept au cas des nœuds relation : l'équation 1.1, page 29 peut être enrichie avec un poids numérique $\alpha \in [0,1]$, conduisant à :

$$\begin{array}{ll} (r,\alpha) & -0-[c:i] \\ & -1-[c':i'] \end{array} \tag{2.7}$$

Le poids peut être interprété comme représentant la compatibilité des nœuds concept c et c' définis par une fonction propre à la relation r qui renvoie α.

Pour l'exemple considéré, on peut par exemple construire :

$$\begin{array}{ll} (\textit{assister}, 0.8) & -0-[\textit{Étudiante} : \textit{Nouka}] \\ & -1-[\textit{Histoire} : \textit{1H008}] \end{array}$$

ce qui peut représenter le fait que l'étudiante *Nouka* a assisté à *la plupart* des cours d'histoire *1H008*. Une interprétation alternative peut être que *Nouka* a assisté *rêveusement* au cours *1H008*, sans être totalement concentrée.

Poids en tant que modalité linguistique

De même, Wuwongse et Cao, 1996 proposent d'appliquer le principe discuté dans la section 2.2.2, page 50 pour les nœuds concept au cas des nœuds relation, c'est-à-dire de

pondérer les nœuds relation par une modalité linguistique définie sur K :

$$\begin{array}{ll}(r,\lambda) & -0-[c:i] \\ & -1-[c':i']\end{array} \tag{2.8}$$

où chaque modalité linguistique $\lambda \in K$ correspond à un sous-ensemble flou de $[0,1]$.

Poids au niveau ontologique

Enfin, le cas des poids au niveau ontologique formalisé par un treillis flou, tel que présenté dans la section 2.2.3, page 54 pour les types de concept, est appliqué au cas des types de relation par Cao et al., 1997 :

$$\begin{array}{ll}(r_\lambda) & -0-[c:i] \\ & -1-[c':i']\end{array} \tag{2.9}$$

où λ est une variable linguistique dans K comme introduit précédemment.

Du point de vue du graphe conceptuel flou, la relation r_λ est un type classique de relation appartenant à un treillis particulier de types, un treillis de types de base associés aux variables linguistiques dans K. En un sens, comme dans le cas des concepts, un tel graphe conceptuel flou est un graphe conceptuel classique avec un vocabulaire particulier.

2.5.2 Pondération de la description d'un nœud concept

La description d'un nœud concept est un champ comprenant un graphe conceptuel ou un ensemble de graphes conceptuels permettant de décrire ce nœud. (Sowa, 2008) Elle est similaire à la définition de graphes conceptuels imbriqués ou stratifiés (Chein & Mugnier, 1997 ; Croitoru et al., 2005).

Morton, 1987, d'après Wuwongse et Manzano, 1993, utilise une définition de nœuds concept formant des triplets constitués d'un type c, d'un marqueur m et d'une description d. Cette description est elle-même un graphe conceptuel qui représente le nœud concept. Cette notion est disponible dans le formalisme de Sowa, 1983, et est similaire à celle des graphes conceptuels imbriqués (Chein & Mugnier, 2008), restreinte à un seul niveau d'imbrication.

Morton généralise cette formalisation des graphes conceptuels et inclut un poids de la description d qui s'écrit :

$$[c:m,(d,\alpha)] \tag{2.10}$$

La valeur α représente la compatibilité entre c et d, et est, comme les autres définitions de Morton, modélisée par une fonction de compatibilité. Wuwongse et Manzano précisent que la compatibilité ainsi définie par Morton peut aussi être modélisée par une

fonction de $[0,1]$ vers $[0,1]$ représentant alors une compatibilité avec les valeurs de vérité dans $[0,1]$. Cette compatibilité ne concerne que la partie factuelle des graphes conceptuels telle qu'elle est définie pour chaque couple type-description, et les descriptions en elles-mêmes sont des graphes conceptuels qui ne sont pas, a priori, définis dans la partie ontologique.

2.5.3 Règle floue

Une règle-λ dans le formalisme des graphes conceptuels, également appelée règle d'inférence, est une extension classique des graphes conceptuels rappelé dans le chapitre 1, permettant les raisonnements par inférence (Chein & Mugnier, 2008) : pour deux concepts c et c', une relation r, un marqueur individuel i et une variable $*x$, une telle règle peut être écrite :

$$[c : *x] \Rightarrow [c : *x] - 1 - (r) - 0 - [c' : i] \tag{2.11}$$

Elle est interprétée comme une règle de la forme *SI un sous-ensemble du graphe conceptuel correspond à l'hypothèse* (par exemple, dans l'équation ci-dessus, le graphe conceptuel considéré contient *x de type c*), *ALORS le graphe conceptuel peut être étendu et/ou spécialisé pour correspondre à la conclusion* (par exemple, dans l'équation ci-dessus *x est en relation avec i de type c' via la relation r*). On peut envisager des règles avec des prémisses et des conclusions plus complexes.

Par exemple, on peut avoir la règle :

$$[Histoire : *x] \Rightarrow [Histoire : *x] - 1 - (apprendre) - 0 - [Étudiante : *]$$

qui représente l'élément de connaissance : *SI x est une leçon d'histoire, alors il existe une étudiante qui apprend la leçon x*.

Wuwongse et Cao, 1996 présentent les *programmes* de graphes conceptuels flous et Cao et Creasy, 2000 le *développement* de graphes conceptuels flous-λ et de graphes conceptuels flous universellement quantifiés, qui utilisent des quantificateurs universels, notés $\forall$, au lieu de marqueurs génériques. Ces propositions, même si elles n'y font pas explicitement référence, peuvent être utilisées comme des extensions des règles-λ de sorte qu'elles autorisent des poids flous dans la prémisse et dans la conclusion.

Dans le premier cas (Wuwongse & Cao, 1996), elles sont la transposition directe des règles-λ au cas des graphes conceptuels flous ayant des types flous au niveau des concepts et des relations, et des valeurs floues au niveau de la valeur d'un type attribut. Ainsi, l'hypothèse et la conclusion de la règle sont de tels graphes conceptuels flous. Ces inférences spécifiques sont une transposition des règles de déduction définies pour les graphes conceptuels flous (Wuwongse & Cao, 1996).

Dans le second cas (Cao & Creasy, 2000), l'opération de *développement* des graphes conceptuels flous-λ et des graphes conceptuels universellement quantifiés met ces graphes conceptuels flous spécifiques sous une forme différente, qui correspond à une règle-λ. Par exemple, si l'on considère le graphe conceptuel universellement quantifié :

$$\begin{array}{rl} (assister) & -0-[\acute{E}tudiante:\forall] \\ & -1-[Histoire:1H003] \end{array}$$

qui représente *Toutes les étudiantes ont assisté au cours d'histoire 1H003*, son extension, qui correspond comme indiqué ci-dessus à une règle λ, s'écrit :

$$\begin{array}{rcl} [\acute{E}tudiante:*x] & \Rightarrow & \\ & \begin{array}{rl} (assister) & -0-[\acute{E}tudiante:*x] \\ & -1-[Histoire:1H003] \end{array} & \end{array}$$

qui représente la même information sous une autre forme. Le cas flou est alors celui où le quantificateur $\forall$ est par exemple remplacé par un ensemble flou sur $[0,1]$, représentant un quantificateur générique. Cette valeur représente dans quelle mesure on peut déduire la conclusion à partir de la prémisse représentée, par exemple avec la valeur *Plupart* qui donnerait *La plupart des étudiants ont assisté au cours 1H003*.

2.6 Bilan

La discussion comparative présentée dans ce chapitre montre la richesse et la diversité des propositions permettant de modéliser les connaissances imprécises dans le cadre des graphes conceptuels flous : ces dernières enrichissent le modèle classique des graphes conceptuels et augmentent leur expressivité et interprétabilité en intégrant des composantes floues à différents niveaux avec des interprétations et des utilisations variées.

Comme nous l'avons vu dans ce chapitre, il s'agit de nœuds concept flous, de nœuds relation flous, de types flous, de marqueurs flous et de valeurs floues, ainsi que de règles d'inférence floues. Une vision complémentaire est offerte dans le tableau 2.1, page 42 qui propose une taxonomie des emplacements de connaissances imprécises, en se concentrant sur la distinction entre les parties ontologique et factuelle, détaillant les variations particulièrement riches des extensions floues des connaissances ontologiques. Il met en évidence les distinctions opérées dans ce chapitre pour formaliser tous les modèles de graphe conceptuel flou proposés et discute les diverses interprétations de ces modèles, mettant en évidence qu'au-delà des cas spécifiques pour lesquels ils ont été définis, ils offrent des outils originaux pour représenter les informations imprécises.

Les perspectives offertes concernent l'étude les conséquences de ces choix sur les processus d'inférence de raisonnement proposés par les articles qui ont introduit ces extensions de graphes conceptuels flous, en fonction de l'interprétation choisie qui peut varier, comme discuté dans ce chapitre. Le cas des connaissances incertaines, qui peuvent s'appuyer sur la même formalisation que les connaissances imprécises discutées dans ce chapitre mais qui sont associées à d'autres outils d'interprétation et de raisonnement, présente un intérêt particulier.

Deuxième partie

Simulation de connaissances

Comme détaillé en partie I, le formalisme des graphes conceptuels établit un cadre de représentation de la connaissance offrant de nombreux avantages. Il présente cependant l'inconvénient de la difficulté à construire des bases de graphes conceptuels sans expertise du formalisme : il n'existe pas, à notre connaissance, de jeu de données de graphes conceptuels de qualité suffisante qui soit immédiatement accessible. Les jeux de données disponibles sont par exemple de petits modèles illustratifs, sans connaissance ontologique ou alors seulement supportée par des hiérarchies plates, c'est-à-dire sans relation d'ordre définie entre les types. C'est par exemple le cas du jeu de données de graphes conceptuels pour NLP/NLU (Elseidy et al., 2014).

Parallèlement, il existe des jeux de données d'entreprise, ou issus de projets privés, qui sont potentiellement de bien meilleure qualité, mais ils sont inaccessibles ou d'utilisation protégée. Le besoin de disposer publiquement d'une base de graphes conceptuels de qualité n'est pas nouveau. Il a déjà été souligné par la communauté au travers d'articles tels que Baget et al., 2010a et Croitoru et al., 2007 pour l'évaluation.

Cet inconvénient rend difficile la validation d'outils proposés pour exploiter des connaissances représentées dans ce formalisme, comme exprimé pour notre cas en partie III. Les connaissances sous forme de graphes conceptuels ont un intérêt si on dispose d'outils d'exploration, mais le développement de tels outils nécessite de disposer de telles bases de connaissance, à des fins de validation, c'est un cercle vicieux.

La simulation de connaissances, comprise comme la génération de connaissances synthétiques dans un environnement contrôlé, peut permettre d'offrir une solution à ce problème. S'appuyant sur un modèle de données, qui peut prendre la forme d'une distribution sur ces dernières ou bien d'un ensemble de contraintes restreignant leur espace de définition, la simulation construit des connaissances desquelles on peut formuler des résultats attendus pour valider des algorithmes. Par exemple, on peut générer un ensemble d'attaques informatiques par une distribution sur un ensemble d'attaques informatiques prédéfinies ou sur des caractéristiques de ces attaques (actions menées, stratégie utilisée, temps d'exécution, …), et utiliser ces bases générées pour tester un algorithme de détection d'attaque. Ainsi, simuler des attaques, là où attaquer effectivement son propre système est problématique à mettre en place, peut permettre de tester des attaques plus variées et ainsi développer un système d'analyse d'attaques et de défense plus efficace.

A contrario, un des obstacles de la simulation de connaissances est la génération de données trop proches du modèle de données choisi : l'évaluation se trouve limitée aux cas connus ou similaires, un problème également présent dans d'autres domaines tels que dans l'apprentissage automatique par exemple.

Dans cette partie, deux approches de simulation de bases de graphes conceptuels sont successivement examinées dans les deux chapitres. Le chapitre 3 considère d'abord le cas

de la simulation de bases de graphes conceptuels par traduction à partir d'un autre formalisme : le modèle *RDF/RDF(S)*. Des algorithmes de traduction existants sont présentés et sont suivis de la proposition d'extensions pour permettre le traitement de cas ambigus.

Ainsi, les interprétations formulées permettent de mieux exploiter les bases de triplets *RDF/RDF(S)* largement disponibles, dans le but d'obtenir des bases de graphes conceptuels de qualité, tout en minimisant la perte de cohérence et de complétude de la traduction.

Le chapitre 4, page 97 considère ensuite le cas de la simulation de bases de graphes conceptuels par génération à partir de contraintes ontologiques permettant une plus grande variabilité de bases générées. Les critères de validation qui y sont proposés et discutés sont une mesure de cette variabilité, la prédictibilité, la richesse de représentation exploitée et le coût de calcul de la génération.

Chapitre 3

Simulation de connaissances par traduction à partir de *RDF/RDF(S)*

Introduction

Une première approche de construction de graphes conceptuels consiste à partir d'une base de connaissances représentées dans un autre formalisme et à en effectuer une traduction. Ce chapitre s'intéresse à la mise en œuvre de ce principe pour le cas du modèle *RDF/RDF(S)* (Manola et al., 2004 ; Brickley et al., 2014) introduit dans la section 1.2.1, page 23 et brièvement décrit ci-dessous dans la section 3.1.1. La traduction du formalisme *RDF/RDF(S)* en graphes conceptuels peut être effectuée via les algorithmes T_3 et T_{nat} (Baget et al., 2009 ; Baget et al., 2010b) disponibles dans le logiciel CoGui (Baget et al., 2010b ; Buche et al., 2014), qui permet la visualisation et la manipulation de graphes conceptuels.

Le principal critère de validation de ces algorithmes est l'équivalence sémantique entre les raisonnements effectués en *RDF/RDF(S)* avant la traduction et les raisonnements effectués dans le formalisme des graphes conceptuels après la traduction. Leur but est d'assurer que les mêmes conclusions sont déduites des mêmes prémisses dans les deux jeux de données, et que les raisonnements demeurent identiques quand les données en graphes conceptuels sont retraduites en *RDF/RDF(S)*, assurant ainsi que la traduction effectuée est de qualité.

Cependant certains prédicats *RDF/RDF(S)* ne peuvent pas être immédiatement traduits dans le formalisme des graphes conceptuels. En effet, les deux formalismes ne sont pas syntaxiquement équivalents : par exemple, en ce qui concerne la distinction entre connaissances ontologiques et connaissances factuelles. La spécification de *RDF/RDF(S)* n'impose pas qu'une étiquette soit nécessairement soit un marqueur individuel, soit un

concept, soit une relation, au contraire des graphes conceptuels.

Après un bref rappel sur le formalisme *RDF/RDF(S)* et les algorithmes T_3 et T_{nat} dans la section 3.1, ce chapitre propose une discussion des cas ambigus en section 3.2 qui peuvent survenir et des traductions possibles selon les interprétations qui en sont faites : il examine successivement trois interprétations factuelles, une interprétation ontologique et une interprétation syntaxique, en explicitant dans chaque cas la traduction proposée et sa réciproque, les présupposés sur lesquelles elle repose et les conséquences qui en découlent.

3.1 Contexte

Le fragment de *RDF/RDF(S)* pertinent pour ce chapitre est introduit, avec en particulier la présentation des triplets *RDF/RDF(S)* traduits, puis les algorithmes T_3 et T_{nat} sont successivement présentés.

3.1.1 Le modèle *RDF/RDF(S)*

Comme évoqué dans le chapitre 1, section 1.2.1, page 23, *RDF* (Manola et al., 2004), pour *Resource Description Format*, est un langage pour la représentation de prédicats logiques binaires sous forme de graphes étiquetés. Le tableau 3.1, page 80, proposé en partie par Baget et al., 2010b, liste les équivalences immédiates entre un triplet *RDF/RDF(S)*, un graphe conceptuel, qui suit le formalisme des graphes conceptuels basiques avec signature, et une formule logique dans le formalisme de la logique des prédicats du premier ordre.

RDF(S) (Brickley et al., 2014), pour *RDF Schema*, étend le langage *RDF* pour permettre de renseigner des informations terminologiques, telles les signatures de relations et un ordre entre les types. La connaissance constituant la base de connaissances factuelles est le triplet (s, p, o), pour sujet, prédicat, objet, et qui représente s et o mis en relation par p.

D'autre part, il existe les littéraux qui permettent de représenter des valeurs au moyen de nombres ou de chaînes de caractères, et les *blanks* qui sont des entités anonymes que nous notons $*B$. Enfin, *Resource* est le type par défaut d'une base *RDF(S)*, et correspond au symbole $\top$ utilisé comme racine de la hiérarchie de concepts T_C dans le formalisme des graphes conceptuels.

Le tableau 3.1, page 80 est la base d'une approche naïve pour la traduction entre *RDF/RDF(S)* et graphes conceptuels qu'est T_{basic}, également proposé par Baget et al., 2010b, qui consiste simplement à reprendre les équivalences entre les colonnes 1 et 2 du tableau. L'article vise à faire respecter un critère de conservation des raisonnements

avant et après la traduction. Cependant ce critère n'est pas respecté par cette première approche en considérant des variables et des *blanks*, ce dont l'article avait besoin. Or les distinctions claires effectuées dans le formalisme des graphes conceptuels ne permettent pas de prendre en compte, par exemple, une relation et un concept ayant le même identifiant. Les algorithmes de traduction T_3 et T_{nat} répondent à ce problème en proposant deux solutions différentes.

Pour un exemple illustratif lié aux attaques informatiques, une même étiquette, disons *attaque informatique*, peut être mise en jeu dans un prédicat où elle est traitée comme une instance en particulier, par exemple *attaque informatique est une instance d'évènement* et *attaque informatique survient dans mes systèmes informatiques*.

La première assertion en langage naturel peut être formalisée en :

$$(attaqueInformatique, estUn, évènement)$$

qui correspond à une instanciation de *évènement*.

La seconde assertion peut être formalisée en :

$$(attaqueInformatique, survientDans, mesSystèmesInformatiques)$$

Dans les deux prédicats, *attaqueInformatique* est traité comme une entité spécifique.

Un problème rencontré lors de la traduction est qu'une base *RDF/RDF(S)* ne contraint pas l'apparition d'un prédicat mettant en jeu *attaqueInformatique* comme un élément de connaissance ontologique. Par exemple, on peut avoir la formule atomique *attaque informatique est une sorte d'évènement* qui se formalise en

$$(attaqueInformatique, estUneSorteDe, évènement)$$

qui traite cette fois *attaqueInformatique* comme une classe ou un type au même titre que *évènement*.

Dans le contexte où nous avons besoin d'une base de graphes conceptuels, cette transformation T_{basic} pourrait nous suffire, dans un premier temps. Cependant, seules les transformations T_3 et T_{nat} ont été implémentées dans CoGui. T_{basic} offre l'avantage néanmoins de bien introduire les traductions plus complexes qui suivent en offrant une traduction immédiate et facilement compréhensible.

3.1.2 T_3

Principe

T_3 (Baget et al., 2009) est une traduction qui privilégie la conformité des raisonnements. Pour ce faire, les triplets *RDF(S)*, constitués chacun d'un sujet, d'un objet, et d'un

RDF/RDF(S)	Graphe conceptuel	Logique
C *rdf* : *type rdfs* : *Class*	type de concept C	prédicat unaire C
R *rdf* : *type rdf* : *Property*	type de relation binaire R	prédicat binaire R
C *rdfs* : *subClassOf* D	$C \preceq D$	$\forall\ (C(x) \implies D(x))$
R *rdfs* : *subPropertyOf* S	$R \preceq S$	$\forall x \forall y (R(x,\ y) \implies S(x,\ y))$
R *rdfs* : *domain* C	$\sigma(R) = (C,\ -)$	$\forall x \forall y (R(x,\ y) \implies C(x))$
R *rdfs* : *range* D	$\sigma(R) = (-,\ D)$	$\forall x \forall y (R(x,\ y) \implies D(y))$
$m_1\ R\ m_2$	R -0- $[\top : m_1]$ -1- $[\top : m_2]$	$R(m_1,\ m_2)$
$*B\ R\ m$	R -0- $[\top : *]$ -1- $[\top : m]$	$\exists x (R(x,\ m))$
m *rdf* : *type* C	nœud concept $[C : m]$	$C(m)$

TABLE 3.1 – Équivalence des principaux prédicats *RDF/RDF(S)* avec les graphes conceptuels et les formules logiques, proposée par BAGET et al., 2010b. Les trois dernières lignes ont été ajoutés pour compléter

prédicat, sont représentés par un nœud relation vide relié à trois nœuds concepts correspondant respectivement aux trois éléments du triplet *RDF(S)*. Chacun des éléments d'un triplet (s, p, o) est ainsi traduit en un nœud concept respectivement de marqueur s, p et o autour d'un nœud relation générique de type *triple* (d'arité 3). Une hiérarchie de types de concepts est définie selon certains triplets *RDF(S)*, comme indiqué dans les six premières lignes du tableau 3.1.

Exemple

Ainsi (*attaqueInformatique*, *estUn*, *évènement*) est traduit par :

$$\begin{aligned}(\textit{triple}) \quad & -0 - [\textit{URI} : \textit{attaqueInformatique}] \\ & -1 - [\textit{URI} : \textit{estUn}] \\ & -2 - [\textit{URI} : \textit{évènement}]\end{aligned}$$

où *URI* est un identifiant unique utilisé dans les langages du web sémantiques, tel que décrit en section 1.2.1, page 23.

Propriétés

T_3 est une traduction correcte et complète par rapport aux raisonnements en *RDF(S)*, mais n'est pas intuitive visuellement, comme illustré ci-dessus. T_R est réduit à une unique

relation, $T_R = \{triple\}$. La représentation graphique est donc alourdie et bien moins lisible. De plus la sémantique des nœuds relation en tant que liens entre entités au sein des graphes conceptuels n'est plus représentée.

Un autre inconvénient (Baget et al., 2010b) est que T_3 ne constitue pas des bases de graphes conceptuels structurant la connaissance, notamment par une distinction stricte entre connaissances factuelles et ontologiques. En effet, les connaissances ontologiques, ainsi que les connaissances factuelles, traduites d'une base *RDF/RDF(S)* par T_3 sont systématiquement traduites dans la partie factuelle des graphes conceptuels. La seule modification du vocabulaire est l'ajout de marqueurs individuels.

3.1.3 $\mathbf{T_{nat}}$

T_{nat} est une seconde traduction qui tente de combler les deux principaux manques de T_3.

Principe

T_{nat} (Baget et al., 2010b) exploite la séparation entre les connaissances ontologiques et les connaissances factuelles en traduisant les prédicats comme des nœuds relation binaires reliant sujet et objet, tous deux traduits comme nœuds concept.

Si une entité est traitée comme deux sortes de connaissances à la fois, l'entité est considérée comme ambiguë et différents choix sont faits en fonction de la situation : si une violation de cette exigence de séparation entre classes et propriétés se produit, l'ensemble de la base *RDF/RDF(S)* à traduire est rejeté ; si une violation se produit entre classes et instances, ou propriétés et instances, les triplets impliquant l'entité ambiguë comme instance sont ignorés (et ceux l'impliquant comme classe, respectivement propriété, sont conservés).

Une particularité des bases de données de graphes conceptuels constituées avec T_{nat} est que seules des relations d'arité 2 sont construites, en raison des restrictions *RDF(S)* qui ne représentent que des relations d'arité 2. Cet inconvénient est minimisé par le fait qu'une relation d'arité supérieure à 2 peut toujours être ramenée à un ensemble de relations d'arité 2, et inversement. Ceci est immédiat si l'on considère que les graphes conceptuels sont des représentations graphiques des formules de la logique des prédicats du premier ordre et que les relations correspondent à des formules atomiques, qui sont 2-décomposables (Jeavons et al., 1998).

Propriétés

Cette traduction assure trois propriétés qui permettent une meilleure représentation des graphes conceptuels mais limitent l'équivalence des raisonnements.

Premièrement, la *séparabilité* est assurée, c'est-à-dire la distinction claire entre les différente sortes de connaissances, en particulier ontologiques et factuelles. Alors que le langage *RDF/RDF(S)* ne s'en préoccupe pas, T_{nat}, lors de la traduction, impose la *condition de séparabilité* à l'ensemble des triplets *RDF/RDF(S)* traduits. Cette condition stipule que toute entité de la base de connaissances apparaît soit comme une classe, une propriété ou une instance (au sens de *RDF/RDF(S)*). Cette propriété implique que T_{nat} se limite au fragment des triplets *RDF/RDF(S)* compatibles avec la condition.

Deuxièmement, elle est *naturelle* dans le sens où les classes sont traduits en concepts, les propriétés en relations et les entités en individus (et les *blanks* en marqueurs génériques).

Enfin, la troisième propriété assurée par T_{nat} est que, en conséquence des deux propriétés précédentes, les connaissances traduites profitent de la représentation intuitive des graphes, contrairement à T_3.

Problèmes d'ambiguïté

Un problème à l'utilisation de T_{nat} vient du fait que certains triplets *RDF/RDF(S)* comportent une ambiguïté du point de vue du formalisme des graphes conceptuels. En effet, un élément dans *RDF/RDF(S)* peut à la fois faire référence à un élément ontologique, tel un type de concept, et à un élément factuel, tel un marqueur individuel. Un tel élément x, c'est-à-dire une étiquette ou identifiant, est bien unique mais peut être mis en jeu dans un triplet en tant qu'élément factuel, par exemple "x est une instance de C", et dans un autre triplet en tant qu'élément ontologique, par exemple "x est un sous-type de C".

Le tableau 3.2, page 83 donne la liste des triplets *RDF(S)* pris en compte par T_{nat} en effectuant la distinction selon que la traduction soit dans le vocabulaire ou dans les graphes conceptuels.

T_{nat} opère un traitement des triplets ontologiques d'abord, puis des triplets factuels.

Enfin, $T_{nat}-$ est la traduction inverse également proposée par Baget et al., 2010b, qui transforme une base de graphes conceptuels en triplet *RDF/RDF(S)*. Elle est à considérer tant elle permet d'illustrer la cohérence et la complétude de T_{nat} en vérifiant que suite à l'utilisation de la traduction inverse $T_{nat}-$, on obtient la base *RDF/RDF(S)* d'origine ou une équivalente.

	Triplet *RDF/RDF(S)*	Traitement par T_{nat}
Ontologiques	$(C,\ rdf:type,\ rdfs:Class)$	Ajout du type de concept C à T_C
	$(R,\ rdf:type,\ rdf:Property)$	Ajout du type de relation R à T_R
	$(C,\ rdfs:subClassOf,\ D)$	Ajout des types de concept C et D à T_C avec $C \preceq D$
	$(R,\ rdfs:subClassOf,\ S)$	Ajout des types de relation R et S à T_R avec $R \preceq S$
	$(R,\ rdfs:domain,\ C)$	Ajout du type de relation R à T_R, du type de concept C à T_C, et spécialisation de la signature de R
	$(R,\ rdfs:range,\ D)$	Ajout du type de relation R à T_R, du type de concept D à T_C, et spécialisation de la signature de R
Factuels	$(i,\ rdf:type,\ C)$	Ajout du nœud concept $[c:i]$ à un graphe conceptuel, du marqueur individuel i à I, et du type de concept C à T_C
	$(*B,\ rdf:type,\ C)$	Ajout du nœud concept $[C:*]$ à un graphe conceptuel, et du type de concept C à T_C
	$(s,\ p,\ o)$	Ajout des nœuds concept et relation correspondants et du type de relation p à T_R

Table 3.2 – Traitement des triplets *RDF(S)* par T_{nat}

3.2 Cas ambigus et leurs interprétations

Dans la suite, nous proposons, en allant d'une présentation assez informelle à une présentation de plus en plus formelle, un ensemble d'interprétations permettant de traiter les ambiguïtés relevées lors de l'utilisation de la traduction T_{nat}. Ces interprétations proposées ont pour but d'atteindre une cohérence des raisonnements au prix de plus ou moins de présupposés à admettre.

3.2.1 Notation textuelle et conventions

Nous introduisons la notation textuelle, partiellement reprise de l'article présentant T_3 et T_{nat} (Baget et al., 2010b), utilisée ici pour représenter des triplets *RDF(S)* dans le tableau 3.2, page 83 ainsi que dans la suite du chapitre. Nous introduisons également nos conventions pour discuter des ambiguïtés à traiter.

Nous considérons un élément ambigu, noté x, à la fois traité comme une classe et une instance (respectivement un type de concept et un individu au sens du formalisme des graphes conceptuels).

À titre d'exemple, nous considérons les trois triplets suivants :

$$n_1 = (x,\ rdf:type,\ rdfs:Class)$$

$$n_2 = (x, p_1, o_1)$$

$$n_3 = (x, p_2, o_2)$$

Le premier triplet n_1 illustre le cas où x est traité comme une classe, tandis que les deux suivants n_2 et n_3 traitent x comme une instance et permettent d'illustrer les traitements différents effectuées à deux instances de x, selon l'interprétation proposée.

Chacune d'elles consiste en la proposition d'une manière d'interpréter la présence de ces triplets dans la même base minimisant la perte de cohérence et de complétude de la traduction. L'ambiguïté à lever est que x est d'une part traité comme une classe, et donc un ensemble d'entités, et comme un individu, et donc comme une entité en particulier.

Cette section présente les 5 interprétations que nous proposons pour traiter les cas considérés comme ambigus par la traduction T_{nat}. Nous commençons par présenter la démarche adoptée lors de la description puis la discussion de chacune des interprétations ainsi que les motivations d'une telle proposition. Nous poursuivons par une présentation successive des 5 interprétations selon trois catégories : trois factuelles où les faits décrits dans n_1 et n_2 référencent des entités de type x, une ontologique où les faits traitent de toute entité de type x, et une syntaxique qui attribue l'ambiguïté à une confusion syntaxique, comme l'homonymie par exemple.

3.2.2 Principes et motivations

Nous nous limitons au cas d'une ambiguïté entre individu et classe.

Chacune des interprétations remplace les triplets ambigus par d'autres, et par exemple réécrit les triplets n_1, n_2 et n_3 définis ci-dessus. Chaque interprétation est détaillée comme suit :

Tout d'abord une présentation informelle donne le principe général de la proposition afin d'en avoir une compréhension intuitive. Ensuite, une description précise de la manière dont nous proposons d'interpréter l'ambiguïté est donnée : pour chaque interprétation, nous fournissons la traduction correspondante, la transformation inverse, et nous explicitons les présupposés qui résultent du choix d'interprétation.

Enfin, nous terminons par une analyse prenant du recul sur les implications d'un tel choix d'interprétation à partir d'un exemple et des présupposés.

3.2.3 Interprétations factuelles

Les interprétations que nous qualifions de factuelles et que nous détaillons tour à tour ci-dessous, correspondent au cas où x est considéré comme une classe et que ses références dans les nœuds n_2 et n_3 sont vues comme représentant soit un même individu

(interprétation unifiée), ou deux individus différents (interprétation prudente). Un troisième choix conserve et représente l'ambiguïté sur ces entités (interprétation unifiée par coréférence).

Cette catégorisation provient du fait qu'il coexiste deux types d'ambiguïtés. Premièrement l'ambiguïté propre à l'interprétation par T_{nat}, qui ne peut pas traiter la cooccurrence de triplets traitant x comme classe et de triplets traitant x comme individu. Elle est résolue par les trois premières interprétations par un choix dit d'interprétation factuelle : chacune fait le choix d'affirmer que x est un type comme le propose n_1, et que les triplets factuels n_2 et n_3 font référence à une ou plusieurs entités de type x. Deuxièmement, la seconde ambiguïté découle de la première ambiguïté : toutes ces entités référencées par x sont-elles une unique entité, des entités différentes, ou bien existe-t-il une imprécision sur cette unicité ? Les trois interprétations factuelles se distinguent sur le traitement de cette seconde ambiguïté.

Nous présentons d'abord l'interprétation factuelle unifiée, suivie de l'interprétation factuelle prudente et de l'interprétation unifiée par coréférence.

3.2.3.1 Interprétation factuelle unifiée

Cette interprétation discute du cas où les entités dans n_2 et n_3 sont un même individu non identifié de type x : elle équivaut à les remplacer par les triplets $n_4 = (*B,\ p_1,\ o_1)$, $n_5 = (*B,\ p_2,\ o_2)$ et $n_6 = (*B,\ rdf : type,\ x)$.

Elle se justifie par l'intuition que les triplets factuels font référence à x pour spécifier une instance non identifiée ou quelconque d'une classe d'URI x. Cette propriété est nommée unification des triplets factuels car les traitements proposés consistent lors de la traduction à la mise en correspondance de toutes les instances de x en individu à un unique nœud concept $[x : *]$.

Traduction n_1 est traduit par l'ajout du type de concept x.

Les triplets n_4 et n_5 quant à eux sont traduits par l'ajout des nœuds concept $c_1 = [Resource : *]$, $c_2 = [Resource :\ o_1]$ et $c_3 = [Resource :\ o_2]$, ainsi que des nœuds relation $r_1 = (p_1)$ et $r_2 = (p_2)$ et leurs liens respectifs pour la partie factuelle. Pour la partie ontologique, n_4 et n_5 entraînent également l'ajout des marqueurs individuels o_1 et o_2 et des types de relation p_1 et p_2 dans le vocabulaire.

Enfin le triplet n_6 produit l'affectation du type x au nœud c_1, ce qui donne : $c_1 = [x : *]$.

Traduction inverse La traduction par $T_{nat}-$ de la base constituée des connaissances obtenues par la précédente traduction nous donne les triplets ontologiques suivants :

Triplet RDF	$T_{nat} \longrightarrow$ graphes conceptuels	$T_{nat}- \longrightarrow$ RDF
n_4	$c_1 = [Resource : *]$ & $c_2 = [Resource : o_1]$ Ajout du type de relation p_1 et du marqueur o_1 Ajout du nœud $r_1 = (p_1)$ connexe à c_1 et c_2	n_4 $(p_1, rdf : type, rdf : Property)$
n_5	$c_3 = [Resource : o_2]$ Ajout du type de relation p_2 et du marqueur o_2 Ajout du nœud $r_2 = (p_2)$ connexe à c_1 et c_3	n_5 $(p_2, rdf : type, rdf : Property)$
n_6	$c_1 = [x : *]$ Ajout du type de concept x	n_6 $(x, rdf : type, rdfs : Class)$

TABLE 3.3 – Traduction par T_{nat} puis $T_{nat}-$ d'un cas ambigu grâce à l'interprétation factuelle unifiée

$(x, rdf : type, rdfs : Class)$, $(p_1, rdf : type, rdf : Property)$ et $(p_2, rdf : type, rdf : Property)$. Les triplets factuels traduits sont les triplets : $(B, rdf : type, x)$, (B, p_1, o_1) et (B, p_2, o_2).

Les deux traductions successives sont réunies dans le tableau 3.3, page 86. Il est à noter que pour la traduction inverse ainsi que pour ce tableau, les triplets axiomatiques indiquant par exemple que o_1 est de type *Resource* ne sont pas renseignés car toujours vrais.

Présupposés Cinq présupposés ont été établis pour cette interprétation.

Le premier présupposé est que x est bien une classe, comme le suggèrent les triplets ontologiques. Ses corollaires sont que les triplets ontologiques sont vrais et que les triplets factuels sont imprécis en utilisant l'URI x.

Le deuxième présupposé est que x est instancié. Ainsi il existe bien au moins un individu de classe x et les triplets factuels font référence à un de ces individus.

Le troisième présupposé est que x n'est pas l'URI d'un individu. Alors que le langage *RDF/RDF(S)* n'opère pas de distinction, ce présupposé impose donc que x ne peut pas identifier un individu.

Le quatrième présupposé est que les instances identifiées par x dans cette base sont une seule entité non identifiée. En effet, l'unification ne concerne que les triplets factuels où un individu est identifié par l'URI x (c'est bien un individu dans notre interprétation). Les autres individus de type x dans la base sont bien a priori distincts de ceux compris dans l'unification. Ces autres individus sont ceux impliqués dans un triplet de la forme $(i, rdf : type, c)$ où $c = x$ ou bien où $c \leq x$, c'est-à-dire c est une spécialisation de x.

Le cinquième et dernier présupposé est que l'individu identifié par x est de type x.

C'est un des présupposés à la base de l'intuition de cette interprétation, car il justifie que l'utilisation de x dans un triplet factuel n'est pas une erreur, a un but bien précis, et que ce but est de référencer une entité non identifiée de type x.

Discussion Cette interprétation correspond, en reprenant l'exemple illustratif des attaques informatiques pour enrichir les triplets n_1, n_2 et n_3 :

$$(attaqueInformatique,\ rdf : type,\ rdfs : Class)$$

$$(attaqueInformatique,\ niveauDanger,\ dangereuse)$$

$$(attaqueInformatique,\ dateÉvènement,\ 22/12/12)$$

.

De tels triplets factuels semblent faire référence à une attaque informatique en particulier, de par l'utilisation de la même URI *attaqueInformatique*, chaque triplet donnant une information sur une caractéristique différente de l'attaque. D'autre part, les types de relations utilisés *niveauDanger* et *dateÉvènement* semblent se compléter. Il en résulte que les associer au même individu semble un choix raisonnable si on rencontre ces triplets dans une base.

Cette interprétation, en regardant le résultat de $T_{nat}-$, permet de retrouver les triplets *RDF/RDF(S)* d'origine (à un renommage près des triplets factuels avec individu non identifié). La traduction semble ainsi cohérente.

Cependant, tout dépend justement du présupposé 4 qui propose l'unification des individus d'URI x. C'est le présupposé le plus important, dans le sens où s'il est faux l'interprétation perd beaucoup de sa cohérence. Dans le cas général où les raisonnements ne tiennent pas en compte du présupposé 4 et ne traitent que ce qu'il y a dans les bases, à savoir les triplets *RDF/RDF(S)* d'une part, et le vocabulaire et les graphes conceptuels d'autre part, la cohérence de cette interprétation n'est pas garantie.

La complétude est cependant assurée par la conservation des triplets par les deux traductions successives comme résumé dans le tableau 3.3, page 86.

L'unification (présupposé 4), c'est-à-dire le fait de faire correspondre les instances factuelles d'URI x à un unique individu, tend à justifier l'unicité de x, c'est-à-dire le fait que la même URI x ait été utilisée pour tous les triplets factuels considérés. Autrement dit, l'unification donne une justification ontologique à ce type d'ambiguïté.

3.2.3.2 Interprétation factuelle prudente

La deuxième interprétation que nous proposons est similaire à la première sur deux points : x est considéré comme étant l'URI d'une classe, et les triplets factuels font référence à une entité non identifiée de type x. Elle se distingue par le refus de l'unification

de ces entités. Ainsi, nous proposons avec cette interprétation que les entités référencées par x dans les triplets factuels sont potentiellement différentes. Il en résulte lors de la traduction autant de nœuds concepts génériques $[x : *]$ que de triplets factuels ambigus.

Le triplet n_5 est alors remplacé par $n_7 = (*B', p_2, o_2)$ et on ajoute le triplet $n_8 = (*B', rdf : type, x)$. L'URI blank utilisée $*B'$ dans n_7 et n_8 se démarque ainsi de l'URI blank $*B$ dans n_4 et n_6.

Traduction Le résultat est similaire à celui de l'interprétation factuelle unifiée : la différence est que pour la partie factuelle, au lieu d'ajouter un seul nœud concept $[x : *]$, nous en ajoutons deux connectés respectivement aux nœuds relation p_1 et p_2.

Traduction inverse La différence avec l'interprétation factuelle unifiée est également dans la partie factuelle : on obtient avec l'exemple considéré les triplets $(*B, rdf : type, x)$, $(*B', rdf : type, x)$, et on retrouve n_4 et n_7.

La distinction entre B et B' s'opère car il existe deux nœuds $[x : *]$ à traduire, donc a priori différents : alors que le langage *RDF/RDF(S)* distingue ses entités génériques par un identifiant "blank" différent, le formalisme des graphes conceptuels les distingue en créant deux nœuds concepts génériques distincts.

En effet, en reprenant le tableau 3.3, page 86, à deuxième ligne, n_5 devient n_7, et donc T_{nat} résulte en l'ajout d'un nœud supplémentaire $c_4 = [Resource : *]$, spécialisé lors de la traduction de n_8 en $c_4 = [x : *]$.

Présupposés Les présupposés sont identiques à ceux de l'interprétation factuelle unifiée, le quatrième non inclus.

On peut se limiter aux quatre présupposés similaires et préciser que l'interprétation factuelle prudente ne fixe rien sur l'unification des occurrences factuelles de x.

Discussion En reprenant l'exemple, l'interprétation factuelle prudente peut s'illustrer par le cas où on disposerait des triplets :

$$(attaqueInformatique, rdf : type, rdfs : Class)$$

$$(attaqueInformatique, dateÉvènement, 22/12/12)$$

$$(attaqueInformatique, dateÉvènement, 23/12/12)$$

Les deux triplets factuels cette fois semblent bien référencer deux occurrences d'attaque informatique différentes. Il existe une possibilité que ce soit la même attaque repérée à deux instants différents, mais excepté le fait que chacun utilise l'URI *attaqueInformatique*, rien ne permet de l'affirmer. Ainsi cette interprétation permet de traiter un tel exemple là

où l'interprétation factuelle unifiée résulterait potentiellement en une perte de cohérence lors de la traduction.

Cette interprétation, comme on peut le voir avec la traduction inverse, ne permet pas de retrouver les triplets d'origine. Dans la spécification *RDF*, il est précisé que les marqueurs blanks n'ont qu'une portée locale, et que c'est au système qui définit et utilise ces marqueurs de définir une politique quant à leur potentielle unicité ou pas. Ainsi, sauf spécifié autrement, dans les conventions *RDF/RDF(S)*, un marqueur générique fait potentiellement référence à la même entité qu'un autre marqueur générique d'identifiant différent.

Or dans le formalisme des graphes conceptuels, c'est également le cas : deux nœuds concepts génériques différents font potentiellement référence à la même entité anonyme. Les traitements issus de cette interprétation entraînent donc une perte de continuité entre les occurrences de x en tant qu'individu, contrairement au cas précédent. Le seul lien les unissant encore est leur type commun x et leur caractère générique.

On en déduit que la différence de résultat entre la base *RDF/RDF(S)* d'origine et la base résultante de la traduction inverse consiste en une perte de connaissance, mais il n'y a pas d'ajout de connaissance. La connaissance perdue est que les triplets factuels référençant x n'utilisent plus la même plus la même URI suite à la traduction inverse : chacun contient une URI blank différente.

Cela dépend en effet de l'interprétation des blanks, qui comme le précise la spécification *RDF* (Manola et al., 2004), dépend de la base qui les utilise.

Ainsi, cette interprétation résulte en une perte de complétude mais pas de cohérence.

Une première manière de compenser cette perte de complétude est de se placer dans l'hypothèse du monde clos, impliquant que tout identifiant différent référence, si non précisé, une entité différente. Une seconde manière de compenser cette perte de complétude est de garder dans la base de graphes conceptuels traduite un lien entre les instances de type x. Cette seconde manière est utilisée dans l'interprétation suivante.

3.2.3.3 Interprétation factuelle unifiée par coréférence

La troisième interprétation que nous proposons est une solution intermédiaire entre les deux premières interprétations proposées. Elle consiste toujours à considérer que x est l'URI d'une classe et que les triplets factuels l'utilisant font référence à une ou plusieurs entités anonymes de type x. Elle propose par contre d'étendre le fragment des graphes conceptuels considéré par T_{nat} dans Baget et al., 2010b en considérant les classes de coréférence des graphes conceptuels simples tels que présentés dans Chein et Mugnier, 2008, Chapitre 3, mais avec un traitement particulier propre à l'interprétation.

Dans le contexte de cette interprétation nous proposons de définir des graphes concep-

tuels simples particuliers où la classe de coréférence ne porte pas le même sens que pour le cas classique : là où dans le cas classique une classe de coréférence regroupe un ensemble de marqueurs référençant la même entité dans la réalité, nous proposons de définir des classes de coréférence qui regroupent des marqueurs référençant *potentiellement* la même entité. L'imprécision sur ces classes de coréférence relève plus d'une indétermination : il y a en effet une potentielle unification, ce qui est renseigné par la présence de la classe de coréférence.

Traduction Les traitements pour la traduction sont similaires au cas de l'interprétation factuelle prudente. Les mêmes connaissances sont générées dans la base de graphes conceptuels, à la différence qu'une classe de coréférence ici est définie entre tout marqueur générique issu de la résolution d'une telle ambiguïté.

Ainsi partant des nœuds concepts génériques $c_1 = [x : *]$ et $c_2 = [x : *]$ issus de de l'ambiguïté (il peut en exister d'autres, traduits de triplets tels que $(B, rdf : type, x)$ par exemple), nous définissons la classe de coréférence $Coref_x = \{c_1, c_2\}$.

Traduction inverse La traduction inverse $T_{nat}-$ ne prend pas en compte les classes de coréférence, donc il n'y a aucune différence avec la précédente interprétation. La modifier en prenant en compte les classes de coréférences résulte en la possibilité de choisir entre les traitements de la première interprétation en unifiant les occurrences de la classe sous l'URI x, ou bien les traitements de la deuxième interprétation en différenciant ces occurrences.

Un troisième choix peut être effectué pour résoudre l'ambiguïté et reporter son traitement ultérieurement : créer une relation de coréférence par la définition d'une propriété p^x_{coref} (au sens de *RDF/RDF(S)*) pour chaque classe de coréférence, qui met en relation deux à deux les éléments de la classe. La définition d'une classe (au sens de *RDF(S)*) risque cependant de mener à des problèmes de compatibilité, car il faudrait alors s'assurer qu'elle soit dans la hiérarchie de classes de l'ontologie, puis qu'elle soit compatible avec x.

Présupposés Cette interprétation part des mêmes présupposés que pour l'interprétation factuelle unifiée, avec la différence que le présupposé 4 devient : les instances de x identifiées par x dans cette base sont potentiellement une seule entité non identifiée. L'information est préservée par la classe de coréférence $Coref_x$ dans la base de graphes conceptuels, et par la propriété p^x_{coref} dans la base *RDF/RDF(S)*.

Discussion Cette interprétation, si on regarde la traduction inverse, conserve la cohérence des connaissances selon le traitement choisi. De plus elle conserve la complétude,

car en prenant en compte cette interprétation dans les raisonnements, aucune information n'est ni perdue ni modifiée par l'utilisation de la classe de coréférence. Ainsi, quel que soit le traitement choisi, la prise en compte de cette interprétation dans les raisonnements assure la cohérence et la complétude de ces raisonnements.

Une extension de cette interprétation est possible pour permettre de représenter plus subtilement l'ambiguïté : par l'ajout d'une pondération associée à chaque classe de coréférence, on peut ainsi quantifier l'ambiguïté selon le cas rencontré. Ainsi, en reprenant l'exemple illustratif, selon que l'on dispose des triplets factuels

$$(attaqueInformatique, niveauDanger, dangereuse)$$

$$(attaqueInformatique, dateÉvènement, 22/12/12)$$

ou bien des triplets factuels :

$$(attaqueInformatique, dateÉvènement, 22/12/12)$$

$$(attaqueInformatique, dateÉvènement, 23/12/12)$$

on peut définir deux degrés de compatibilité différents. En effet, dans le second cas, comme discuté dans la deuxième interprétation, il est peut-être moins probable que dans le premier cas que les sujets des deux triplets factuels fassent référence à la même attaque informatique. Ce raisonnement peut par ailleurs être étendu à d'autres ensembles de triplets *RDF/RDF(S)*.

3.2.4 Interprétation ontologique

Une quatrième interprétation que nous proposons, dite ontologique, détermine x comme un type également, mais généralise les triplets factuels comme n_2 et n_3 comme représentant toute instance de type x.

Nous proposons dans cette quatrième cas de modifier un des choix effectués par les trois interprétations précédentes. x est toujours considéré comme étant l'URI d'une classe, cependant ses occurrences dans les triplets factuels sont interprétées différemment. En effet, les connaissances traitant x comme instance sont ici interprétées comme étant des connaissances générales (ou implicites) sur x. Ainsi, nous considérons ici que l'ensemble des occurrences de l'URI x font nécessairement référence à la classe x.

Afin de traduire ces connaissances générales (ou implicites) sur x, le choix est fait de les traduire en règles-λ.

x est ici interprété comme une classe et non un individu. D'autre part les connaissances factuelles sur x sont à portée générale sur tous les individus de type x, d'où l'appellation "ontologique".

Traduction

De la même manière que pour les précédentes interprétation, le type de concept x est ajouté dans le vocabulaire ainsi que les types de relation p_1 et p_2 et les marqueurs individuels o_1 et o_2.

Pour la partie factuelle, les nœuds concept $[Resource : o_1]$ et $[Resource : o_2]$ sont ajoutés.

Enfin, dans l'ensemble Λ des règles-λ, les deux règles-λ :

$$[x : *y] \implies [x : *y] - 0 - (p_1) - 1 - [Resource : o_1]$$

$$[x : *y] \implies [x : *y] - 0 - (p_2) - 1 - [Resource : o_2]$$

sont ajoutées, où les $[x : *y]$ sont les nœuds de correspondance de la règle.

Traduction inverse

$T_{nat}-$ ne gère par les règles-λ donc il faut ajouter un traitement spécifique à cette interprétation.

Afin de retranscrire les règles-λ issues de la résolution d'ambiguïté, on génère pour chaque règle-λ le triplet factuel qui lui correspond, en réutilisant x comme URI.

Présupposés

Le premier présupposé est modifié. Il consiste toujours en l'affirmation que x est un classe. Cependant, ses corollaires sont légèrement modifiés par cette nouvelle interprétation : les triplets ontologiques et factuels ne sont plus ambigus, ils sont tous vrai. En outre, il n'y a plus de contradiction, et les triplets factuels ne sont plus imprécis en utilisant l'étiquette x pour désigner un individu, ils représentent en fait une vérité générale sur les instances de x.

Le troisième présupposé est repris tel quel, x dans cette interprétation n'est en effet pas l'URI d'un individu.

Enfin, un nouveau présupposé est admis pour cette interprétation : les occurrences de x en individu font référence à la classe x en tant que connaissance générale sur tous les individus de type x.

Discussion

Ces interprétations offrent un nouvel angle sur le sens que portent les triplets traitant x en tant qu'individu.

Le premier et dernier présupposé sont importants. Il faut réfléchir à la perte de cohérence entre un triplet factuel *RDF/RDF(S)* qui devient dans le contexte de cette interprétation une connaissance générale vraie sur toute la base.

Prenons d'abord l'exemple illustratif. Considérons les triplets ontologiques :

$$(attaqueInformatique, rdf : type, rdfs : Class)$$

et les triplets factuels

$$(attaqueInformatique, niveauDanger, dangereuse)$$

$$(attaqueInformatique, dateÉvènement, 22/12/12)$$

Les triplets factuels sont interprétés par les règles générales suivantes, transcrites en langage naturel : "Toute attaque informatique est dangereuse" et "Toute attaque informatique est survenue le 22/12/12". La première règle peut avoir du sens si la propriété a un ensemble de définition plus large que la seule URI "attaqueInformatique". Sinon, elle ne pourrait avoir que pour objet la valeur "dangereuse". La seconde règle peut avoir du sens dans le contexte où la base considérée est la récupération d'informations sur un système d'information sur une période, et que cette règle est déduite de l'analyse de ces données.

Pour justifier une telle interprétation il faut se poser la question de l'utilisation de triplets *RDF/RDF(S)* conjointement avec cette interprétation ontologique, et surtout du but d'un tel choix.

Cette interprétation entraîne ainsi une perte de cohérence certaine, car il est peu probable que les triplets factuels traitent de savoir général, mais elle reste complète. Elle a, au delà de ces critères de cohérence et de complétude, un intérêt pour la simulation de connaissances, permettant de générer une base de graphes conceptuels avec règles-λ à partir d'une telle base *RDF/RDF(S)* sans que ce soit trop artificiel : la base ainsi générée voit ses règles-λ comme généralisation de faits ambigus.

Une manière d'ajouter en subtilité sur cette interprétation est de proposer une interprétation alternative, où la dualité classe-individu propre au langage *RDF/RDF(S)* de x n'est pas tranchée. Ainsi, chaque URI ambiguë est considérée parfois comme une classe, parfois comme un individu, et parfois comme une notion intermédiaire. Cela peut se traduire par la génération de règles-λ ou de graphes conceptuels référençant ni x en tant que classe, ni x en tant qu'individu, ni une instance de type x, mais par exemple un ensemble pondéré sur le domaine de x, comme sous-ensemble de $T_C \times T_R \times M$.

3.2.5 Interprétation syntaxique

Enfin une dernière interprétation considère que x dans n_1 et x dans n_2 et n_3 ne font pas référence au même élément de la réalité, et que la même URI découle d'un problème syntaxique tel que l'homonymie.

Cette cinquième interprétation se base sur l'idée que l'utilisation de x en tant que classe d'une part, et en tant qu'individu d'autre part est une erreur d'homonymie : malgré l'unicité, c'est peut-être celle qui fait le moins de présupposés. Elle peut induire néanmoins une perte d'information systématique. Elle considère qu'il y a homonymie entre éléments de différentes catégories, et règle les ambiguïtés en renommant toutes les entités en leur ajoutant un suffixe relatif à leur catégorie. Par exemple, le traitement pourrait être, à partir d'un objet x ambigu, de renommer ses occurrences en tant que type de concept et d'individu (et de type de relation) respectivement en $x\#C$ et $x\#I$ (et $x\#R$).

Ainsi, les triplets n_1, n_2 et n_3 choisis deviennent respectivement :

$$(x\#C,\ rdf : type,\ rdfs : Class)$$

$$(x\#I,\ p_1,\ o_1)$$

$$(x\#I,\ p_2,\ o_2)$$

À un renommage près, la traduction est cohérente et complète. C'est peut être l'interprétation la plus prudente car elle garde toute l'information.

L'ambiguïté classe-entitée résolue, on peut évoquer le second cas d'ambiguïté : l'unification ou non des $x\#I$. Si l'on s'en tient à l'explication syntaxique, on a deux choix.

En premier lieu on peut présupposer que non seulement les $x\#I$ en tant qu'individus sont des homonymes de $x\#C$ la classe, mais également que les $x\#I$ sont des homonymes, et donc ne référencent pas le même individu.

Cette approche prudente consisterait alors à renommer tous les $x\#I$ en $x\#I_0$, $x\#I_1$, … voire à également renommer les $x\#C$ de façon similaire dans une optique encore plus prudente.

En second lieu, on peut plutôt présupposer, comme pour le cas de l'interprétation ontologique, que les connaissances sur les $x\#I$ sont des connaissances sur un même individu ou groupe d'individus, possiblement confondus avec les $x\#C$.

Du point de vue du modèle *RDF/RDF(S)*, les $x\#I$ et $x\#C$ restent la même URI x : l'ambiguïté n'est pas un problème dans ce formalisme. Du point de vue des graphes conceptuels, les $x\#I$ et $x\#C$ sont respectivement des individus et des types de concepts. Ainsi à toute connaissance du type, "J'ai la connaissance p sur $x\#I$ qui est une entité" exprimée en graphes conceptuels, la traduction inverse en *RDF/RDF(S)* sera cohérente. En effet, la dualité classe-entité qui y règne inclut à la fois la notion d'entité et la notion de classe, et donc la phrase est comprise comme "J'ai la connaissance p sur x qui est une entité, et une classe également, de toutes façons la différence n'est pas effectuée.".

Ainsi cette interprétation permet une cohérence des résultats partant de cet exemple et des triplet choisis.

3.3 Bilan

Nous avons dans ce chapitre présenté des algorithmes de traduction depuis une base *RDF/RDF(S)* vers une base de graphes conceptuels. Nous avons ensuite proposé différentes manières de résoudre des cas ambigus non gérés par les algorithmes de traduction. Ces différentes interprétations ont une portée qui va au delà de la traduction, tant elles permettent de mieux saisir l'interaction entre le langage *RDF/RDF(S)* et le formalisme des graphes conceptuels, de représenter des connaissances de manière plus variée et subtile, et surtout de poser une base pour la simulation de connaissance, en particulier la simulation de connaissance à partir d'une traduction.

Pour accomplir une simulation de connaissances à partir des interprétations de cas ambigus, il faudrait à partir d'une même base d'entrée se munir de paramètres ou d'un procédé stochastique pour permettre de générer des connaissances proches. Par exemple, dupliquer des connaissances et les attribuer à des URI compatibles, comme le triplet :

$$(attaqueInformatique, dateÉvènement, 22/12/12)$$

qui pourrait être modifié en changeant le littéral de la date *22/12/12* ou en spécialisant l'URI *attaqueInformatique* à un type d'attaque en particulier.

Une autre idée est de profiter de l'ambiguïté d'interprétation, qui offre plusieurs possibilités de génération, pour construire des connaissances sur le domaine d'interprétation. Bien sûr, dans un tel cas, il faut veiller à conserver un degré seuil de cohérence pour ne pas générer des connaissances absurdes. Selon les garanties formelles, explicitées dans notre proposition, que l'on veut assurer à la traduction, une interprétation différente pourra être choisie.

Cela fait partie des problématiques notamment abordées avec le chapitre qui suit sur la simulation à partir de contraintes ontologiques.

Chapitre 4

Simulation de connaissances par génération à partir de contraintes ontologiques : *CG2A*

Introduction

La simulation de connaissances par traduction présentée dans le chapitre précédent est tributaire des bases utilisées : celles-ci assurent un degré de réalisme aux bases générées. Néanmoins, la contrepartie est que les bases traduites limitent la variabilité des bases générées et que la richesse du formalisme des graphes conceptuels n'est pas totalement exploitée.

Ce chapitre propose de s'intéresser à une seconde approche : la simulation de connaissances sur la base d'un ensemble de contraintes. Ces dernières constituent un modèle sous-jacent de données, et définissent l'espace des graphes conceptuels à généraliser. Ce chapitre s'attelle de plus à permettre d'identifier des connaissances a priori des bases générées. Pour cela, les méthodes proposées se basent sur un type particulier de contraintes pour la simulation : des contraintes ontologiques dont le modèle de données prend la forme de connaissances ontologiques au sein, notamment, d'un vocabulaire.

Le savoir ontologique forme ainsi un modèle sous-jacent du jeu de données généré. L'avantage est que l'utilisateur dispose de connaissances explicites sur tous les jeux de données générés depuis ce modèle, sans avoir besoin de les analyser. Ce principe est inspiré par la validation expérimentale mise en œuvre dans la communauté du clustering par exemple, où des jeux de données artificiels sont générés depuis une distribution fournie, qui permet de définir un résultat de partition attendue pour un algorithme appliqué à ces données.

En terme de processus stochastique, on peut considérer un ensemble de graphes conceptuels en tant que variables sur un espace de graphes conceptuels prédéfinis. Les graphes conceptuels générés sont constitués par affectation de variables, c'est-à-dire la construction d'un graphe conceptuel suivant les contraintes de l'espace prédéfini, puis la combinaison de ces graphes conceptuels. On peut par exemple noter Γ un tel espace, dont chaque élément est un graphe conceptuel à structure et étiquettes prédéfinies $G = \{C, R, E, label\}$. Pour accroître encore la variabilité, on peut également définir au sein de chaque graphe conceptuel un ensemble de variables à la place de certaines étiquettes : chacune d'elles devient ainsi un ensemble des possibles, prédéfini dans les contraintes ontologiques du vocabulaire, et affectées lors de la génération. Ces variables suivent une loi d'un ensemble de variable aléatoires de C et R vers les espaces de définition $T_C \times M$ et T_R respectivement.

Ces contraintes constituent alors un ensemble d'informations sur la structure, les étiquettes, les motifs récurrents et les caractéristiques globales des bases résultantes.

Ce chapitre présente l'algorithme *CG2A* (Conceptual Graphs Generation Algorithm), que nous avons proposé pour générer une base de graphes conceptuels depuis un ensemble de contraintes ontologiques. La connaissance factuelle est générée depuis cette connaissance ontologique, exprimée par un vocabulaire et un ensemble de graphes conceptuels-γ, c'est-à-dire des éléments de Γ comportant des variables comme introduit ci-dessus. Ce sont des éléments de représentation que nous proposons pour étendre le formalisme des graphes conceptuels.

Les paramètres d'entrée sont d'abord présentés en section 4.1. Ensuite dans la section 4.2, page 100 l'algorithme et son fonctionnement sont introduits. Un ensemble d'extensions proposant, au moyen de procédés stochastiques, la génération automatique de contraintes ontologiques sont présentées en section 4.3, page 103. Enfin une étude expérimentale pour évaluer *CG2A* sur les critères choisis est menée en section 4.4, page 106.

Nos contributions sur la simulation de connaissances par *CG2A* ont été publiées dans les actes de la conférence IFSA-EUSFLAT 2021 (Faci et al., 2021a).

4.1 Paramètres d'entrée

Nous considérons deux types de paramètres d'entrée. D'une part nous détaillons en section 4.1.1 les contraintes ontologiques : elles déterminent la sémantique et la structure des bases générées.

D'autre part, nous présentons en section 4.1.2, page 99 les paramètres numériques permettant de contrôler des caractéristiques globales des bases générées, notamment les conditions d'arrêt de *CG2A*.

4.1.1 Contraintes ontologiques

CG2A prend en entrée un vocabulaire $\mathcal{V}$ et un ensemble de graphes conceptuels-γ $\mathcal{G}$. Ces derniers sont une proposition décrite ci-après.

Le vocabulaire $\mathcal{V} = \{T_C, T_R, \sigma, I, \tau\}$, tel que présenté formellement dans le chapitre 1, section 1.2.2, page 24, contient une hiérarchie T_C sur les types de concept, une hiérarchie T_R sur les types de relation, un ensemble de signatures de relations défini par la fonction σ, un ensemble de marqueurs individuels I et une fonction de conformité τ qui renseigne le type associé à chaque marqueur.

L'ensemble $\mathcal{G}$ constitue l'ensemble des composants des graphes conceptuels générés : les graphes conceptuels-γ. Ce sont des graphes conceptuels particuliers que nous proposons, inspirés par les graphes conceptuels-λ du formalisme des graphes conceptuels (Chein & Mugnier, 2008). Nous les utilisons pour formaliser la notion de variables disposant d'un domaine au sein d'un graphe conceptuel. Par rapport à un graphe conceptuel classique, tel que décrit dans la section 1.2.3, page 28, certaines étiquettes sont remplacées par des variables faisant référence à un ensemble d'éléments du vocabulaire. Cet ensemble constitue le domaine de la variable. Les graphes conceptuels-γ sont donc des contraintes paramétrables.

Un graphe conceptuel-γ $\Gamma = ((v_1, D_1)\dots(v_n, D_n))G$, $n \geq 1$ est un graphe conceptuel G avec n variables v_i et leurs domaines respectifs D_i. Chaque variable v_i est affectée à une étiquette de G, soit une étiquette de type relation, une de type concept, ou une de marqueur. Une représentation est dans la figure 4.6, page 106 où v_1, v_2 et v_3 sont respectivement affectées à un type de concept, un marqueur et un type de relation. Pour une variable v_i associée à un type de relation t_r, son domaine D_i est un sous-ensemble de T_R réduit aux types de relation de même arité, c'est-à-dire $D_i = \{t \in T_R, arity(t) = arity(t_r)\}$. Pour une variable v_i associée à un type de concept t_c, son domaine D_i est un sous-ensemble de T_C réduit aux types de concept respectant toutes les contraintes imposées par les signatures des nœuds relation connectés. Pour une variable v_i associée à un marqueur m_i, le domaine est un sous-ensemble de I réduit aux marqueurs de types de concept identiques ou plus spécifiques, c'est-à-dire $D_i = \{m \in I, type(m) \leq type(m_i)\}$.

Ainsi, $\mathcal{G}$ est construit sur le support $\mathcal{V}$ qui détermine les espaces de définition des graphes conceptuels-γ.

4.1.2 Contraintes numériques

CG2A prend de plus 3 paramètres numériques en entrée : *nbGC*, le nombre de graphes conceptuels à générer, *minSize*, la taille minimale de chaque graphe conceptuel généré en nombre de nœuds, et *maxSpe*, le nombre maximal de spécialisations opérées sur chaque étiquette de type. Plus précisément, la taille de chaque graphe conceptuel généré aug-

mente jusqu'à atteindre au moins *minSize*, puis les étiquettes de type sont spécialisées un nombre de fois tiré aléatoirement entre 0 et *maxSpe*, et le processus se répète avec un autre graphe conceptuel jusqu'à atteindre *nbGC* graphes conceptuels dans la base de sortie. Ces trois paramètres déterminent donc les conditions d'arrêt de l'algorithme.

4.2 Algorithme *CG2A*

Après une vue d'ensemble de *CG2A* en section 4.2.1, son fonctionnement est détaillé en section 4.2.2 par la présentation de son pseudo-code, puis en section 4.2.3 avec un opérateur de fusion de graphes conceptuels.

4.2.1 Principe général

Tout d'abord, *CG2A* génère un graphe conceptuel en combinant de manière aléatoire des graphes conceptuels-γ en entrée jusqu'à atteindre une taille minimale donnée. Ensuite, les variables se voient attribuer des valeurs aléatoires issues de leurs domaines respectifs. Enfin, les étiquettes de types sont spécialisées un nombre de fois donné et les nœuds des graphes conceptuels générés ayant le même marqueur individuel sont fusionnés pour augmenter la connexité du graphe conceptuel résultant. *CG2A* itère jusqu'à ce qu'un nombre *nbGC* de graphes conceptuels ait été généré.

4.2.2 Pseudo-code

La figure 4.1, page 101 donne le pseudo-code de *CG2A*, commenté ci-dessous. L'opérateur *Join* du pseudo-code est noté $\uplus$.

Soit $G_c = (C_c, R_c, E_c, label_c)$ le graphe conceptuel en cours de génération et :

$$\Gamma = ((v_1, D_1)\dots(v_n, D_n))G = (C, R, E, label)$$

un graphe conceptuel-γ de l'ensemble $\mathcal{G}$.

Tout d'abord les variables v_i de Γ sont instanciées avec des valeurs tirées aléatoirement de leurs domaines respectifs D_i, et celles affectées à une étiquette de type sont spécialisées un nombre de fois entre 0 à *maxSpe*, tiré aléatoirement de T_C ou T_R si plusieurs possibilités de spécialisation existent.

Ensuite $G_o = (C_c \uplus C, R_c \cup R, E_c \uplus E, label_c \uplus label)$ est formé à partir de la fusion de G_c et de G, où $\uplus$ est une union proposée basée sur la fusion de nœuds coréférents (Chein & Mugnier, 2004). Sa différence est la suivante : s'il existe des éléments de C_c avec un marqueur individuel identique à celui de C, seul le plus spécialisé est conservé. Elle est détaillée dans la section suivante.

FIGURE 4.1 – Pseudo-code de CG2A

$\mathcal{V} \leftarrow (T_C, T_R, \sigma, I, \tau)$
$\mathcal{G} \leftarrow \{G_i\}$
$nbGC, minSize, maxSpe$
Initialiser $\mathcal{G}_o$ à un ensemble vide
while $size(\mathcal{G}_o) \leq nbGC$ **do**
    Initialiser $G_c = (C_c, R_c, E_c, label_c)$ à un graphe conceptuel vide
    while $size(G_c) \leq minSize$ **do**
        Tirer $((v_1, D_1) \ldots (v_n, D_n))G_i = (C, R, E, label)$ de $\mathcal{G}$
        Affecter une valeur de D_i à chaque variable v_i
        Spécialiser chaque variable d'étiquette de type entre 0 et $maxSpe$ fois
        $G_c = Join(G_c, G_i)$
    end while
    Ajouter G_c à $\mathcal{G}_o$
end while
Renvoyer $\mathcal{G}_o$

Les briques élémentaires, c'est-à-dire une relation et son voisinage, tel que décrit section 5.2.2, page 130, sont fusionnées : tous les nœuds des deux briques élémentaires sont connectés au nœud résultant, c'est-à-dire que les éléments de E_c et E correspondant à deux nœuds fusionnés sont réaffectés au nœud résultant.

4.2.3 Opérateur de fusion

L'opérateur de fusion, noté *Join* dans le pseudo-code, est représenté figure 4.2, page 102, où la couleur des nœuds représente les marqueurs qui leur sont associés et leur motif représente leur type : les deux nœuds verts, respectivement à l'extrémité droite du graphe conceptuel courant et au sommet du graphe conceptuel ajouté, sont fusionnés. Ainsi, l'opérateur fusionne les nœuds de même marqueur. Ils ne sont pas nécessairement de même type ; le type le plus spécifique est retenu. Dans l'illustration, cette étape correspond au nœud résultant qui garde les hachures de l'un des deux nœuds fusionnés.

Une étape supplémentaire, de conformité, vérifie que les étiquettes sont en accord avec les contraintes des signatures. En itérant de relation en relation, les types des nœuds concept connexes sont comparés à la signature et modifiés pour devenir conformes. Cette étape implique une spécialisation de l'étiquette de type au type conforme à tous les

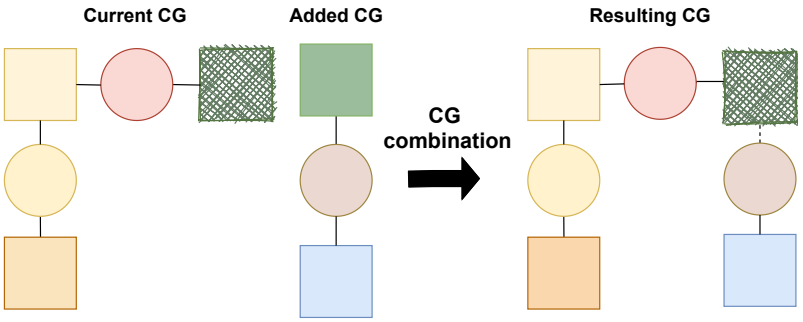


FIGURE 4.2 – L'étape de fusion de graphes conceptuels dans *CG2A*. Cette représentation représente les nœuds concept par des carrés et les nœuds relation par des ronds.

nœuds relation connexes.

Sans l'opérateur $\uplus$, l'algorithme obtiendrait pour chaque GC généré un ensemble d'éléments de $\mathcal{G}$ instanciés non connexes. La connexité des graphes conceptuels résultants dépend ainsi du nombre de nœuds communs. Il existe d'autres techniques de fusion de graphes basée sur l'opérateur de jonction (LAUDY et al., 2007 ; CHEIN & MUGNIER, 2014), mais ce simple opérateur $\uplus$ basé sur l'opérateur de fusion de nœuds coréférents est suffisant dans notre cas.

En effet, afin de tester l'algorithme *cgSpan* d'extraction de motifs fréquents, présenté en chapitre 5, page 119, nous avons besoin de *plusieurs* graphes *connexes* d'une taille suffisamment importante. D'abord car il est basé sur *gSpan* qui ne cherche que les composantes connexes dans plusieurs graphes, ensuite parce que disposer d'une certaine taille de base permet d'obtenir des motifs plus variés et plus intéressants.

CG2A stoppe les combinaisons de graphes conceptuels lorsque la taille minimale souhaitée *minSize* est atteinte, et arrête la génération lorsqu'elle atteint le nombre souhaité de graphes conceptuels générés *nbGC*. Étant donné que des graphes conceptuels constitués de potentiellement plusieurs nœuds sont combinés, les graphes conceptuels qui en résultent sont généralement plus grands que *minSize*. En effet, si on note N_A et N_B le nombre respectif de nœuds des graphes combinés, et si on note N_{AB} le nombre de nœuds en commun, alors suite à la combinaison, la taille du graphe en cours de construction passe de N_A à $N_A + N_B - N_{AB}$. Ainsi, en général la construction stoppe alors que la taille du graphe construit passe d'une valeur strictement inférieure $N_{inf} < minSize$ à une valeur $N_{sup} = N_{inf} + N_B - N_{AB} > minSize$.

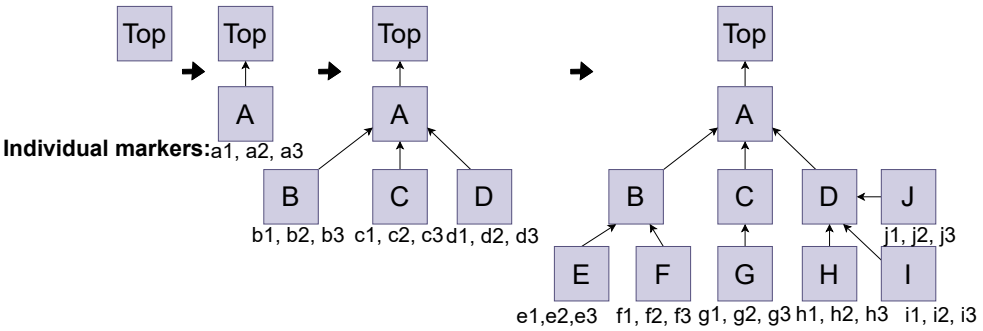


FIGURE 4.3 – Génération automatique d'une hiérarchie de types de concept, ici effectuée en trois étapes. Les paramètres sont : Profondeur= 4 ; Nombre maximum d'enfants= 3 ; Nombre de marqueurs individuels par type= 3.

4.3 Extensions : modules de génération automatique des entrées

Cette sous-section présente trois modules pour générer automatiquement les paramètres d'entrée de *CG2A* afin d'accroître la variabilité et automatiser la génération. Tous les paramètres numériques mentionnés peuvent être remplacés par une moyenne et un écart-type, une valeur est alors tirée de la distribution normale correspondante à chaque utilisation du paramètre.

4.3.1 Génération automatique du vocabulaire

Ce module génère automatiquement le vocabulaire $\mathcal{V}$ selon quatre paramètres : la profondeur respective des hiérarchies des types de concept et de relation, le nombre maximum d'enfants pour chaque nœud de la hiérarchie et le nombre de marqueurs individuels pour chaque type de concept. Cette génération est aléatoire, cependant les quatre paramètres garantissent un nombre de caractéristiques déterminées pour le vocabulaire qui en résulte.

Comme illustré dans les figures 4.3 et 4.4, pour respectivement les concepts et les relations, une structure de hiérarchie est générée jusqu'à atteindre la profondeur désirée, en respectant la contrainte de largeur (le nombre d'enfants). Des étiquettes uniques aléatoires sont attribuées à chaque nœud de la hiérarchie. La hiérarchie de types de concept est un arbre enraciné ayant pour racine le type le plus général "Top", qui correspond à $\top$, comme dans la figure 4.3.

Pour chaque type de concept, une liste de marqueurs individuels est générée.

Pour chaque type de relation, un type maximum est défini pour chaque arité, par exemple noté T_3 pour le cas de l'arité 3 dans la figure 4.4. Ensuite des signatures sont

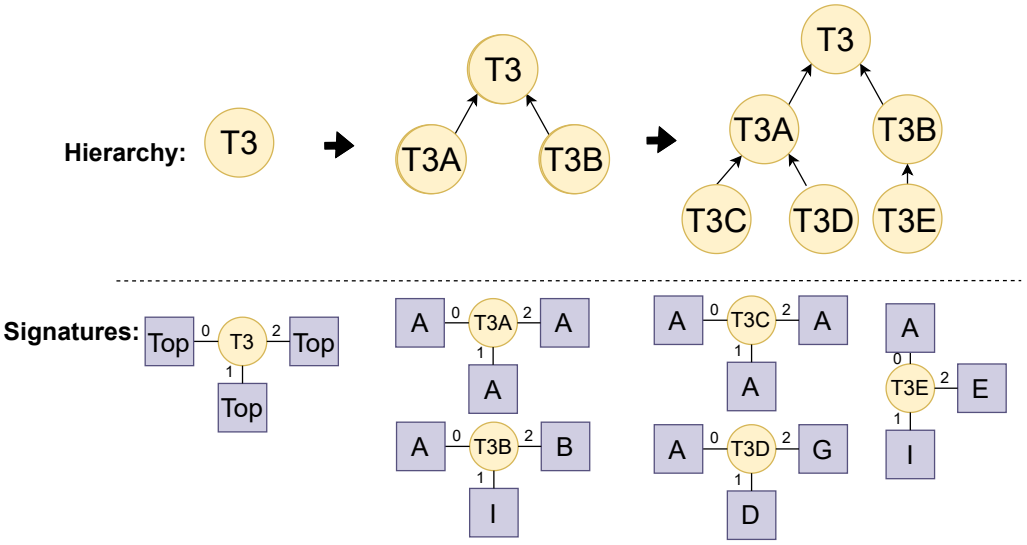


FIGURE 4.4 – Génération automatique d'une hiérarchie de types de relation avec leurs signatures, ici effectuée en deux étapes. Les paramètres sont : Profondeur= 3 ; Nombre maximum d'enfants= 3.

aléatoirement définies pour chaque type de relation en utilisant la hiérarchie de concept précédemment générée. Chaque type de relation maximal a une signature par défaut avec seulement le type de concept "Top" comme contrainte. Un type de relation plus spécialisé qu'un autre a une signature plus restrictive, c'est-à-dire que les restrictions sur les types de concept associées sont identiques ou requièrent des types plus spécifiques. Cette génération de signature est représentée dans la figure 4.4 où à chaque étape la hiérarchie est étendue, puis les signatures sont définies comme identiques ou plus restrictives que les signatures de relations plus générales.

Par exemple, dans l'illustration, les signatures de $T3C$ et $T3D$ sont plus restrictives ou égales à celle de $T3A$, car $T3C$ et $T3D$ en sont une spécialisation. Dans la signature de $T3C$, les contraintes sont identiques à la signature de $T3A$. Par contre la signature de $T3D$ voit l'argument 1 remplacé par D et l'argument 2 remplacé par G. En se référant à la hiérarchie en figure 4.3, G et D sont tous deux plus spécifiques que A qu'ils remplacent.

4.3.2 Génération automatique des graphes conceptuels-γ

Ce module génère automatiquement un ensemble de graphes conceptuels-γ afin d'obtenir l'ensemble $\mathcal{G}$ comme illustré dans la figure 4.5, page 105. Ce module prend en entrée un vocabulaire $\mathcal{V}$ (éventuellement généré automatiquement à l'aide du module décrit dans la sous-section précédente 4.3.1) et deux entiers : le nombre de graphes conceptuels-

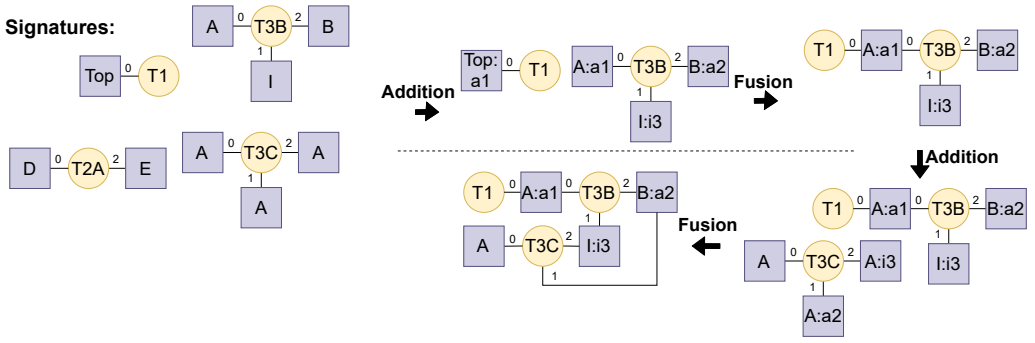


FIGURE 4.5 – Génération automatique de graphes conceptuels-γ

γ à générer et leur taille minimale.

Le module de génération automatique de graphes conceptuels-γ est similaire à l'application de *CG2A* à l'ensemble des signatures extraites du vocabulaire $\mathcal{V}$: il effectue des fusions répétées de signatures. La seule différence est que chaque étiquette est traitée comme une variable sans contrainte. Ainsi, chaque étiquette est attribuée aléatoirement puis est spécialisée aléatoirement.

Les graphes conceptuels-γ générés n'ont ainsi aucune variable définie : il faut dans un second temps les définir manuellement ou les générer automatiquement par l'utilisation du module ci-dessous.

4.3.3 Génération automatique de variables

Ce module génère des variables dans les graphes conceptuels-γ d'entrée. Au lieu de devoir choisir les étiquettes qui sont des variables et leurs domaines respectifs, ce module prend en entrée un ensemble de graphes conceptuels-γ et le vocabulaire correspondant, qui peuvent être générés avec les modules précédents, ainsi que cinq nombres : le nombre de variables associées à un type de concept, à un type de relation et à un marqueur individuel par graphe conceptuel, le nombre de valeurs par variable et le nombre maximal de spécialisations.

Premièrement, pour chaque graphe conceptuel, les variables sont attribuées à un type de relation, à un type de concept ou à un marqueur individuel. Ensuite, une liste de valeurs est associée à chaque variable. La figure 4.6, page 106 illustre cette opération avec les variables v_1, v_2 et v_3 qui correspondent respectivement à une étiquette de type de concept, une étiquette de marqueur et une étiquette de type de relation. Elles sont détaillées ci-dessous.

Les relation sont affectées d'abord, car leur signatures restreignent ensuite les étiquettes des nœuds connexes. Puis les types de concept sont affectés, car ils limitent

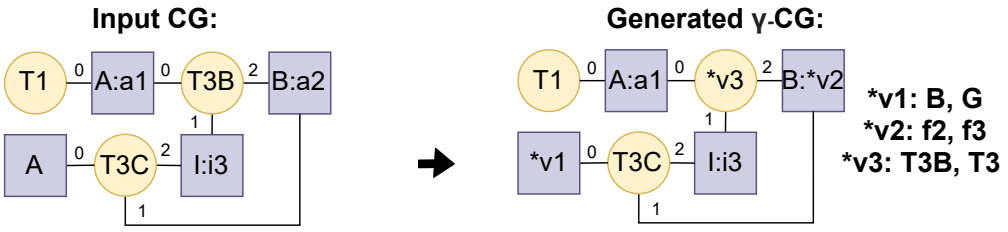


FIGURE 4.6 – Génération automatique de variables

eux-mêmes quels marqueurs peuvent leur être associé. En effet, il faut rappeler qu'un marqueur, dans le formalisme des graphes conceptuels que nous avons choisi, a un type associé par la fonction de conformité τ, et ne peut donc pas être affecté à un type incompatible, c'est-à-dire incomparable par leur relation d'ordre. Par exemple, dans la figure 4.3, page 103, $b1$ est de type B, incompatible avec C, qui ne peut donc pas être associé à $b1$. Pour un type de relation v_3, dans la figure 4.6, page 106, le module choisit parmi les relations de même arité.

Pour un type de concept, comme v_1 dans la figure 4.6, le module choisit parmi les types de concept identiques ou plus spécifiques parmi ceux compatibles avec les signatures des relations connexes.

Pour un marqueur individuel, comme v_2 dans la figure 4.6, il choisit parmi les marqueurs individuels avec un type de concept attribué égal ou plus spécifique que le type de concept du nœud.

Ensuite, toutes les variables affectées correspondant aux étiquettes de type sont spécialisées un certain nombre de fois entre 0 et *maxSpe*.

4.4 Étude expérimentale

Cette section présente l'étude expérimentale menée pour évaluer *CG2A* à la fois en tant que technique de génération de graphes conceptuels, et en comparaison aux techniques de traduction existantes, présentées dans le chapitre 3, section 3.1, page 78. L'entrée utilisée dans ces expériences ainsi que les propriétés souhaitées de variabilité, de prédictibilité, de représentativité et d'efficacité sont présentées dans la sous-section suivante. Les résultats sont ensuite examinés en fonction de chaque critère dans les sections 4.4.2, page 108 à 4.4.5, page 111.

4.4.1 Protocole expérimental

Nous considérons quatre critères à optimiser pour un algorithme de génération de données. Premièrement, l'algorithme doit offrir un degré de *variabilité* des données générées à partir d'une même entrée. En effet, à partir d'un ensemble unique de connaissances ontologiques, la possibilité de produire des ensembles de données de tailles et de caractéristiques différentes est nécessaire pour évaluer l'étendue des connaissances pouvant être simulées.

Ensuite, il faut fournir un certain niveau de *prédictibilité immédiate*. Cela signifie que pour un algorithme de génération de données donné, noté *dGenA*, on peut définir les résultats attendus d'un algorithme d'exploration de données, noté *dMinA*, exécuté sur une base de données générée par *dGenA*. Évidemment, les résultats attendus seront différents en fonction de l'objectif de *dMinA* et de la pertinence des déductions des paramètres d'entrée de *dGenA* par rapport à cet objectif. Par exemple, si *dGenA* n'a qu'un paramètre numérique en entrée, alors peu de déductions pourront être faites sur les bases générées, et il sera alors difficile de formuler des résultats attendus pour *dMinA*. Ce critère est essentiel pour permettre la validation de *dMinA* mais est difficile à quantifier. Cette méthodologie est illustrée dans le cas de l'extraction de motifs fréquents dans la partie III, chapitre 5, section 5.3, page 134.

Troisièmement, l'algorithme de génération doit atteindre une certaine *représentativité*, c'est-à-dire exploiter autant que possible le formalisme des graphes conceptuels. Tout fragment du formalisme qui ne peut pas être représenté limite l'utilisation de l'algorithme de génération à un sous-ensemble de situations où ce fragment est inutile.

Quatrièmement, le *temps de calcul* doit être minimisé. En effet, il peut être crucial d'obtenir rapidement un ensemble de données d'exemples lorsqu'on dispose de connaissances ontologiques. De plus, de nombreuses générations peuvent être nécessaires pour tester différentes variations des paramètres pour satisfaire les attentes.

Ces critères, définis plus formellement dans les sections qui suivent, sont utilisés pour comparer l'algorithme proposé *CG2A* à T_{nat} (Baget et al., 2010b) présenté dans le chapitre 3, section 3.1, avec la configuration suivante. L'entrée de T_{nat} est un jeu de données *RDF/RDF(S)/OWL*, modifié pour résoudre certains problèmes lors de la traduction, en particulier les ambiguïtés décrite dans le chapitre 3, section 3.2, page 83. Le jeu de données modifié comprend une ontologie constituée de 39 types de concept et de 35 types de relation organisés en une hiérarchie de 5 niveaux de profondeur et ayant entre 1 et 3 enfants pour chaque type. Une option proposée par T_{nat} pour diviser les graphes conceptuels en plusieurs composantes connectées a été utilisée, de sorte que chaque graphe conceptuel résultant soit un graphe connexe. Sinon, il en résulte en un seul grand graphe conceptuel non connecté.

Dans les expériences menées, ce découpage a résulté en 18 graphes conceptuels, à partir des 18 composantes connexes du grand graphe conceptuel généré.

L'entrée de *CG2A* correspond aux caractéristiques de la base de données générée avec T_{nat} afin d'assurer que l'on évalue l'influence de la partie algorithmique plutôt que la variabilité induite par le choix des paramètres.

Concernant les contraintes ontologiques, le vocabulaire utilisé est celui renvoyé par T_{nat}. Les graphes conceptuels-γ utilisés en paramètre d'entrée sont quant à eux définis comme des graphes conceptuels renvoyés par T_{nat} dans lesquels on remplace certaines étiquettes par des variables. Leur domaine de définition est cohérent avec le reste du graphe, c'est-à-dire que tout élément du domaine de chaque variable, s'il est affecté à la variable, ne conduit pas à la construction d'un graphe conceptuel incohérent. Lors de l'utilisation des modules, les paramètres d'entrée ont été choisis pour correspondre aux caractéristiques du vocabulaire et des graphes conceptuels-γ définis pour *CG2A*.

Les contraintes numériques sont les suivantes : *nbGC* est ainsi fixé à 18, *minSize* à 30 et *maxSpe* à 2.

L'exécution de T_3 sur le même ensemble de données aboutit à un graphe conceptuel unique d'environ 6000 sommets et un vocabulaire vide (autre que la relation *triple* de T_3). Tous les nœuds de relation sont des nœuds de relation étiquetés du type "triple" connectant les éléments d'un triple RDF/DF(S). Comme cela ne conduit pas à à de nombreux graphes conceptuels ni à une ontologie appropriée, T_3 n'est pas pertinent pour notre besoin de simuler une base de graphes conceptuels avec vocabulaire, et respectant leur biparticité (la seule relation est *triple* avec T_3). Par conséquent, ses résultats ne sont pas utilisés dans ce qui suit.

4.4.2 Résultats sur la variabilité

Afin d'évaluer numériquement la notion de variabilité, nous considérons les critères suivants : la taille moyenne en nombre de nœuds des graphes conceptuels générés, noté *NbN* le nombre moyen d'étiquettes uniques dans un graphe conceptuel, noté *NbL*, tous deux avec leur écart-type et moyenne, et le de relations d'arité de 1 à 3, désigné par *Ar1*, *Ar2* et *Ar3*.

Ces critères sont définis afin de répondre à notre besoin de disposer d'un algorithme de simulation de graphes conceptuels assurant une variabilité au moins sur ces critères. Cependant il est à noter qu'ils ne sont pas optimisés ni même pris en compte par l'algorithme de traduction T_{nat} qui vise à maximiser la conformité de raisonnement entre les bases de données d'entrée et de sortie. En outre, *CG2A* et ses variantes peuvent représenter des relations d'arité plus grande que 3, mais l'utilisation de *Ar1*, *Ar2* et *Ar3* semble suffire pour les expériences présentées.

Test	NbN (avg. ± sd.)	NbL (avg. ± sd.)	Ar1	Ar2	Ar3
T_{nat}	15.2 ± 321	3 ± 1	0	3	0
CG2A	36.3 ± 4	22.5 ± 4	0.5	44	3
Auto Voc	33 ± 3.5	55 ± 14	4	34	9
Auto GC-γ	39.9 ± 2	22 ± 4.1	6	22	31
Auto Var	35.3 ± 4	32 ± 7	0.4	42	7
Full Auto	35 ± 4	67 ± 17	8	33	26

TABLE 4.1 – Caractérisation moyenne des graphes conceptuels générés par une exécution de T_{nat} et différentes versions de *CG2A* dont les résultats sont moyennés sur 100 exécutions (cf. protocole expérimental section 4.4.1, page 107)

La première ligne du tableau 4.1, page 109 donne les résultats de T_{nat}. Les suivantes donnent les résultats moyens de 100 lancements de *CG2A* et ses variantes avec chaque module d'extension individuellement (Auto Voc, Auto GC-γ et Auto Var respectivement) et *CG2A* avec tous les modules d'extension (Full Auto).

La traduction de T_{nat} produit un grand graphe conceptuel comportant plusieurs centaines de nœuds et de nombreux graphes conceptuels de quelques noeuds. C'est pourquoi l'écart-type de *NbN* est significativement plus important que la moyenne de *NbN*, et que le *NbL* est faible. De plus, *Ar1* et *Ar3* sont nuls, ce qui est dû au fait que les langages *RDF/RDF(S)* ne représentent pas les relations d'arités différentes de 2.

CG2A conduit à des écart-types significativement plus petits, et les caractéristiques des graphes conceptuels résultants sont proches des paramètres d'entrée.

Les résultats attendus sont que l'utilisation d'entrées générées automatiquement conduise à une plus grande variabilité dans les résultats attendus. On peut observer que, en effet, l'utilisation d'un vocabulaire aléatoire augmente l'écart-type de *NbL* tandis que l'utilisation de graphes conceptuels aléatoires augmente l'écart-type de *NbN*. Dans les deux cas, il en résulte une plus grande variabilité dans les arités des relations. Les résultats de la méthode entièrement automatisée de *CG2A* combinent ces conséquences.

4.4.3 Résultats sur la prédictibilité

La prédictibilité définit la possibilité de définir a priori les résultats attendus pour algorithme d'exploration de données exécuté sur l'ensemble de données généré. Ainsi, nous nommons *résultats attendus* dans ce qui suit les *résultats ou caractéristiques de résultats en sortie que nous pensons obtenir d'un tel algorithme d'exploration des données*. La prédictibilité peut par ailleurs être mise en perspective avec le coût des ressources spéci-

fiques à déployer et des efforts à entreprendre pour définir les entrées, en particulier les connaissances ontologiques. En effet, plus il y a d'effort à produire des paramètres d'entrée précis, non aléatoires, plus les informations concernant les résultats attendus seront précises.

T_{nat} génère une base de graphes conceptuels à partir d'un jeu de données de triplets *RDF/RDF(S)*, sans nécessiter de connaissances préalables sur ce jeu de données ou son formalisme. Cependant, cela implique que sans explorer le jeu de données d'entrée, T_{nat} ne peut être considéré comme prévisible. En tant que tel, il ne répond pas à l'objectif de prédictibilité immédiate.

CG2A peut être considéré comme permettant des résultats attendus car les graphes conceptuels générés sont définis comme une combinaison des graphes conceptuels-γ d'entrée qui sont définis sur le vocabulaire d'entrée : $\mathcal{G}$ et $\mathcal{V}$ constituent les résultats attendus dont les caractéristiques sont connues.

CG2A utilisé avec la génération automatique de vocabulaire ou de graphes conceptuels-γ change la nature des résultats attendus, qui sont définis en termes de leurs caractéristiques générales plutôt que d'informations spécifiques. Par rapport à *CG2A* avec Auto Voc, seules les caractéristiques générales du vocabulaire sont connues, ses spécificités ne le sont pas. Avec Auto GC-γ, seules les caractéristiques générales des composants utilisés pour construire les graphes conceptuels générés sont connues (nombre et taille des graphes conceptuels-γ, ainsi que les signatures sur lesquelles ils sont construites). Il ajoute à la variabilité de *CG2A* en deux endroits : lors du passage des signatures aux graphes conceptuels-γ d'entrée, puis des graphes conceptuels-γ aux graphes conceptuels générés.

CG2A utilisé avec la génération automatique de variables, Auto Var, ne modifie pas la prédictibilité de manière significative. Nous considérons que les variables automatiques réduisent légèrement la prédictibilité immédiate en augmentant la variabilité.

La variante entièrement automatisée de *CG2A*, qui comprend les trois modules aléatoires, combine leurs propriétés respectives et définit les résultats attendus en fonction de leurs caractéristiques générales. Globalement, les résultats sont significatifs par rapport à T_{nat} : il y a beaucoup plus de variabilité avec *CG2A*, et alors que *CG2A* doit faire face à un équilibre entre variabilité et prédictibilité immédiate, T_{nat} ne permet pas de définir des résultats attendus.

4.4.4 Résultats sur la représentativité

T_{nat} construit des bases de graphes conceptuels n'incluant que des relations d'arité 2, qui sont déséquilibrées en terme de connaissances ontologiques et de connaissances factuelles, c'est-à-dire qui comprennent majoritairement l'une des deux sortes de connais-

sance. Toutefois, l'avantage de disposer de relations d'arités différentes est possiblement une question de perspective ou de reformulation. En effet, toute relation d'arité supérieure à 2 peut être réécrite comme un ensemble de relations d'arité 2 sémantiquement équivalent (Baget et al., 2010b). De plus, il peut également être avancé que le manque d'équilibre entre les deux sortes de connaissance (comme le fait que T_{nat} construit souvent des bases constituées d'un ou deux gros graphes conceptuels, les autres graphes conceptuels ne contenant que quelques nœuds) est principalement dû aux aux caractéristiques des bases *RDF/RDF(S)* considérées en entrée, plutôt qu'à l'algorithme lui-même.

CG2A et ses variantes évitent plus naturellement ces inconvénients. Elles permettent une représentativité de la majeure partie du formalisme des graphes conceptuels tel que rappelé dans le chapitre 1. *CG2A* conserve la plupart des avantages de T_{nat} en utilisant le formalisme de CoGui et ajoute la possibilité de générer une grande proportion de nœuds de relation avec des arités variées, et d'avoir à la fois un large vocabulaire et une quantité considérable de graphes conceptuels, c'est-à-dire des connaissances à la fois ontologiques et factuelles. En outre, lors de la définition des paramètres d'entrée, par exemple les caractéristiques du vocabulaire, l'utilisateur peut déterminer l'étendue du formalisme des graphes conceptuels qui sera exploité. D'une manière générale, *CG2A* garantit que l'utilisateur peut choisir plus précisément les caractéristiques de la base résultante.

4.4.5 Résultats sur l'efficacité

Dans les expériences menées, en fonction des paramètres des conditions d'arrêt, la plupart des exécutions de *CG2A* durent moins d'une seconde et ne dépassent jamais 5 secondes. L'ordinateur utilisé possède 16 Go de mémoire et un processeur i5-2500k. L'utilisation des modules de génération automatique augmente le temps de calcul, d'un facteur 2, cependant le temps passé à concevoir l'entrée sans ces modules n'est pas comptabilisé. La traduction de T_{nat} dure beaucoup plus longtemps, avec un facteur de trois à dix selon la taille de la base d'entrée. A un moment donné, si l'entrée est trop massive, T_{nat} stoppe et les tests ne peuvent être poursuivis.

4.5 Bilan

Ce chapitre a présenté *CG2A*, un algorithme de simulation de connaissances sous forme de graphes conceptuels à partir de contraintes ontologiques.

L'avantage d'utiliser des graphes conceptuels-γ au lieu de définir directement de nombreuses variantes d'un graphe conceptuel est que le processus de construction est plus simple et rapide. De plus un graphe conceptuel-γ conçu, il est possible d'en générer plusieurs tout en conservant sa structure et sa sémantique. D'autre part, lors de

l'évaluation d'un algorithme tel que *cgSpan* qui exploite des connaissances sous forme de graphes conceptuels pour l'extraction de motifs, concevoir des résultats attendus à partir d'un nombre réduit de graphes conceptuels-γ est bien plus aisé.

En conséquence, *CG2A* garantit la variabilité à partir d'une entrée ainsi que la prédictibilité grâce à la connaissance des graphes conceptuels-γ d'entrée et de leurs caractéristiques, voir l'étude en section 5.3, page 134 par exemple où l'utilisation de *CG2A* a permis de formuler des résultats attendus.

CG2A permet en effet une variabilité dans le jeu de données généré plus importante que toute autre méthode connue, car il offre à la fois une grande variabilité dans la taille et les étiquettes des graphes conceptuels ainsi qu'une proportion raisonnable de nœuds de relation avec une arité différente de 2 dans les graphes conceptuels générés. Ainsi, de nombreuses situations différentes peuvent être testées grâce à l'utilisation de *CG2A*, soit un domaine fortement contraint pour tester un cas spécifique, soit une génération plus flexible pour tester une grande variété de situations. En outre, le formalisme des graphes conceptuels est bien représenté avec des hiérarchies profondes, une variabilité dans les signatures et différentes arités de nœuds relation. Enfin, lorsqu'on utilise une méthode différente pour obtenir un jeu de données de graphes conceptuels, comme T_{nat}, il n'est pas possible de définir les résultats attendus sans avoir préalablement exploré le jeu de données ou avoir des connaissances préalables sur ce jeu de données.

Les perspectives de *CG2A* sont de générer des graphes conceptuels plus complexes, par exemple des graphes conceptuels imbriqués ou pondérés. Une autre direction envisage une possibilité différente en termes de prédictibilité, exprimée comme une distribution souhaitée sur les paramètres des graphes conceptuels, afin de s'assurer que le jeu de données résultant respecte une telle distribution.

On peut aussi formaliser la simulation de données comme le résultat d'un processus stochastique, la génération alors étant modélisée selon la prise de valeurs d'un ensemble de variables aléatoires dont les lois respectives correspondent au modèle à suivre. Dans l'exemple de la génération d'attaques informatiques, on peut ainsi considérer une attaque comme une variable aléatoire, ou comme un ensemble de variables aléatoires, et les valeurs comme différents types d'attaques composées de différents évènements systèmes, applicatifs et réseaux.

Une autre alternative est de considérer la structure d'un tel graphe conceptuel comme étant variable également. Dans le cas basique comme avec les alternatives, il faut en tous cas s'assurer, lors de l'affectation des variables, du respect des contraintes inhérentes aux graphes conceptuels : les arités des relations doivent être respectées, les signatures également si cette extension est incluse dans le vocabulaire, et surtout il peut y avoir un besoin de conformité sémantique via le respect de certaines contraintes formulées sous forme de

règles-λ.

Malgré cette génération, la conception même de ce modèle de connaissances peut rester compliquée, déplaçant le problème de la difficile conception d'une base de graphes conceptuels. Néanmoins cette entrée peut être générée automatiquement depuis un ensemble réduit de paramètres numériques, grâce à trois extensions de l'algorithme, qui automatisent respectivement la génération du vocabulaire, des graphes conceptuels-γ et des variables des graphes conceptuels-γ.

La perspective devient alors de construire des *benchmarks* selon deux méthodes : (1) Développer sur la base de *CG2A* un outil de simulation permettant de simuler des bases de connaissances de graphes conceptuels, en reprenant des caractéristiques de bases connues associées à une tâche ; (2) Traduire des bases existantes associées à une tâche dans le formalisme des graphes conceptuels pour disposer d'un premier *benchmark* de référence, disponible pour tous et toutes.

Troisième partie

Exploitation de la connaissance

La troisième partie de nos travaux de thèse porte sur l'exploitation des connaissances représentées dans le formalisme des graphes conceptuels qui offre à la fois des avantages formels et sémantiques, par ses propriétés théoriques et sa richesse expressive.

Plus précisément, nous nous somme intéressés à l'extraction de nouvelles connaissances à partir de bases constituées de graphes conceptuels, en considérant la tâche d'extraction de motifs fréquents. De façon générale, celle-ci vise à identifier des répétitions dans les données étudiées, qui constituent des régularités qui permettant de caractériser et de résumer ces données. Ces régularités offrent ainsi une synthèse interprétable des données : l'interprétabilité étant jugée notamment par le degré de concision et de pertinence de leur représentation. À cet effet, plusieurs techniques existent dans l'état de l'art pour permettre de disposer de connaissances interprétables (Shah et al., 2015 ; Martin & Azvine, 2017).

Cette tâche a été abordée pour de nombreux types de données, et nous effectuons un rapide rappel historique de la tâche et ses principaux algorithmes dans le chapitre 5. Ensuite, nous y présentons un algorithme dédié au cas des graphes conceptuels, en montrant comment on peut exploiter leurs particularités pour gagner en efficacité et en pertinence des résultats. Ces propriétés sont étudiées expérimentalement sur deux types de données : d'abord dans le chapitre 5 sur des données simulées par l'algorithme *CG2A* que nous avons présenté dans le chapitre 4, puis sur des données de trajectoires réelles dans le chapitre 6.

Le chapitre 6 présente l'étude d'un cas d'application correspondant à des besoins industriels. Elle a été mise en œuvre pour tester notre algorithme d'extraction de motifs fréquents dans des graphes conceptuels pour un cas métier.

Chapitre 5

Extraction de motifs fréquents dans des graphes conceptuels : *cgSpan*

Une tâche classique d'exploitation de connaissances consiste à les résumer par leurs caractéristiques récurrentes. Formalisées comme des régularités, appelées motifs fréquents, elles sont définies comme des éléments de connaissance qui se retrouvent en plusieurs occurrences dans une base de données. Ainsi ces motifs fréquents les résument en étant en un certain sens représentatifs de ce qu'elles contiennent.

La section 5.2, page 127 présente l'algorithme *cgSpan* que nous proposons pour le cas des connaissances représentées sous forme de graphes conceptuels : il exploite les spécificités de ces dernières pour augmenter l'efficacité de l'extraction de motifs fréquents en termes d'espace mémoire et de temps de calcul ainsi que la pertinence des résultats. La section 5.3, page 134 décrit les expérimentations réalisées pour comparer *cgSpan* aux algorithmes traitant le cas des graphes étiquetés sur des données artificielles générées par *CG2A*.

Les travaux présentés dans ce chapitre ont été publiés dans les actes de la conférence ICAISC 2021 (Faci et al., 2021b).

5.1 La tâche d'extraction de motifs fréquents

Il existe de nombreux ouvrages et articles qui présentent le vaste domaine de l'extraction de motifs fréquents, ou Frequent Pattern Mining, tels que Han et al., 2007 ; Goethals, 2003 ; Aggarwal et al., 2014 ; Aggarwal, 2015.

Cette tâche a d'abord été élaborée pour des connaissances basiques correspondant à des données ensemblistes (Agrawal et al., 1993). Elle a ensuite été généralisée à des cas plus complexes, par exemple des connaissances structurées sous forme de séquences (Pinto

et al., 2001 ; Ferreira & Azevedo, 2005) ou de graphes (Yan & Han, 2002 ; Iyer et al., 2018 ; Dias et al., 2019). Il en résulte un domaine de recherche qui s'enrichit en se complexifiant. Par exemple, alors que pour le cas des données ensemblistes, la question du nombre d'occurrences d'un motif peut être intuitive, pour ces cas plus complexes elle devient moins évidente : il faut alors proposer une formalisation de motif, définissant notamment sa forme et le calcul de son nombre d'occurrences. C'est l'un des points principaux qui différencient les algorithmes d'extraction de motifs existants.

La section suivante propose une vue d'ensemble sur la tâche et établit l'organisation de la suite de la section.

5.1.1 Vue d'ensemble

L'objectif de cette section est de rappeler trois cas de référence sur lesquels nos travaux se basent : l'algorithme classique *Apriori* qui traite des données ensemblistes et est un des fondateurs de la tâche ; l'algorithme *gSpan* qui traite des données représentées sous la forme de graphes étiquetés ; et *DMGM-GSM* dédié au cas des graphes dont les étiquettes sont de plus organisées en une ou plusieurs taxonomies.

Extraction de motifs fréquents

L'extraction de motifs fréquents est une tâche classique de résumé de base de connaissances pour la réduire à ses éléments les plus significatifs, les motifs fréquents, ainsi que rendre sa représentation plus concise. Deux éléments sont utilisés dans les algorithmes effectuant cette tâche : la génération de candidats et le calcul de support. Des alternatives existent, telles que la croissance de motifs (Pattern growth) (Han et al., 2000) et l'extraction d'hyper-structures (Pei et al., 2001). Nous nous concentrons sur la génération de candidats.

La génération de candidats est une technique visant à générer les motifs potentiellement fréquents. C'est ensuite le calcul de support qui définit si ces motifs sont bien fréquents ou non. On définit le support d'un motif comme son nombre d'occurrences : c'est-à-dire le nombre d'instances de la base d'exemples traitée correspondant au motif. Par exemple, pour une base de transactions et un motif correspondant à un ensemble d'éléments U, son support est défini comme le au nombre de transactions qui contiennent tous les éléments de U. Si le support d'un candidat dépasse un certain seuil, appelé seuil de support et donné en paramètre de l'algorithme d'extraction, on le classe comme motif fréquent.

Il est à noter qu'une alternative usuelle est de définir ce support comme la proportion entre le nombre d'instances que le motif recouvre, et le nombre d'instances total. Aussi, il peut être choisi de ne prendre en compte que certaines occurrences selon des critères

fixés, comme par exemple limiter à une fois par instance, ou bien plus d'occurrences, en étant plus souple sur la mise en correspondance avec le motif.

Il existe beaucoup de possibilités de définition de motifs. Micale et al., 2018 en présentent une partie pour le cas des graphes étiquetés. Ils illustrent notamment que lors du calcul du support, il est possible de ne prendre en compte que la structure de ces graphes, ou bien que l'ensemble des étiquettes qu'ils utilisent.

Extraction de motifs fréquents et intéressants

En plus d'utiliser le critère de fréquence, les algorithmes proposés ont au fur et a mesure tenté de se démarquer par l'utilisation conjointe d'un critère traduisant cet intérêt, souvent correspondant à une degré d'informativité. Par exemple, Inokuchi, 2004 utilise un critère de limitation de la sur-généralisation des motifs dans des graphes étiquetés avec taxonomie, pour limiter l'apparition de motifs étiquetés trop peu informatifs.

Différentes utilisations de ces critères ont ensuite été mises en œuvre. Une première proposition a été d'élaguer les résultats, c'est-à-dire de retirer les motifs minimisant le critère d'intérêt, tel que le degré de significativité statistique par exemple. D'autres ont proposé de renvoyer les motifs les plus fréquents en premier, ou bien les plus fréquents et les plus informatifs au regard du critère d'intérêt. Enfin, une dernière manière que nous proposons dans *cgSpan* est d'effectuer une compression des motifs obtenus en élaguant non pas les motifs les moins intéressants, mais des sous-parties des motifs qui manquent d'intérêt. Han et al., 2007 proposent également de formuler l'intérêt sous forme de motifs compressés ou approximatifs. Il est à noter que ces techniques d'extraction de motifs fréquents, l'utilisation de critères supplémentaires, l'élagage, la compression et la fusion des résultats ont pu être utilisés ensemble au sein des mêmes algorithmes. Enfin, l'utilisation de ces différentes techniques, et notamment l'utilisation conjointe de la maximisation du support et d'un critère d'intérêt, a permis aussi d'obtenir des motifs plus expressifs, c'est-à-dire restituant un maximum d'information en un minimum d'éléments. Cela a pu résulter alors en une meilleure expressivité et une meilleure interprétabilité des résultats.

Organisation

Dans cette section nous présentons deux algorithmes sur lesquels nous avons basé nos travaux et qui sont classiques dans la tâche d'extraction de motifs. Tout d'abord l'algorithme *Apriori* (Agrawal & Srikant, 1994; Asaini, 2019) qui est un algorithme classique d'extraction de règles d'association dans des itemsets. Une règle d'association est composée d'une prémisse et d'une conclusion toutes deux étant des motifs fréquents. Elle représente le fait que lorsque la prémisse est présente dans la base, la conclusion l'est également. Une valeur numérique est associée à la règle et traduit la proportion de

cas dans laquelle cette association se vérifie. C'est un algorithme fondateur : une partie des algorithmes d'extraction de motifs fréquents en est une adaptation à un autre format de données ou une version alternative. L'idée principale d'*Apriori* est notamment de construire les motifs candidats de taille n à partir de motifs fréquents de taille inférieure.

Dans un second temps, nous présentons l'algorithme *gSpan* (Yan & Han, 2002), qui est fondateur dans l'extraction de motifs fréquents dans des graphes étiquetés. *gSpan* a notamment comme contribution une réécriture des graphes en entrée en séquences, ce qui mène à une reformulation du problème. On passe en effet d'une extraction de motifs fréquents dans des graphes étiquetés à une extraction de motifs fréquents dans des séquences d'étiquettes.

Nous concluons cette section par l'introduction de la tâche d'extraction de motifs fréquents dans des graphes étiquetés avec taxonomie à travers une présentation de l'algorithme *DMGM-GSM* (Directed MultiGraph Miner - Generalized Subgraph Mining) (Petermann et al., 2017).

5.1.2 Données ensemblistes : *Apriori*

Certains algorithmes d'extraction de motifs fréquents traitent de données ensemblistes ou tabulaires. (Agrawal et al., 1993 ; Mannila et al., 1994 ; Brin et al., 1997b) Ils recherchent des itemsets fréquents dans une base de transactions, où une transaction est un ensemble d'éléments, et un itemset est un motif de transaction sous forme d'ensemble d'éléments fréquents. Par exemple, ces transactions peuvent représenter un ensemble d'articles achetés ensemble ou un ensemble de collaborateurs sur un même projet. De tels algorithmes, en plus des itemsets, peuvent identifier des règles d'association, qui établissent des liens entre les motifs de la forme "Si le motif X est présent, alors le motif Y l'est également". L'utilisation d'*Apriori* a permis de mettre en évidence dans l'exemple des articles de supermarché quels sont les articles fréquemment achetés ensemble.

Apriori est découpé en étapes, où l'étape k correspond à la construction puis la comptabilisation du support des candidats de taille k. L'algorithme a deux principales composantes : la composition de candidats à partir de motifs plus petits et la recherche en largeur. En premier lieu l'algorithme prend l'ensemble des objets possibles, et comptabilise leur support, puis ne garde que les objets dont le support dépasse le seuil de support donné en paramètre. Ensuite, à chaque étape suivante $k+1$, les candidats explorés sont les combinaisons possibles des candidats de support suffisant de taille k. La recherche s'effectue en largeur car à chaque étape tous les candidats ainsi construits de même taille sont explorés avant de passer à l'étape suivante, construisant petit à petit des motifs fréquents de plus en plus gros.

Enfin, par ailleurs, au delà d'*Apriori* et des algorithmes maximisant l'occurrence, il existe comme évoqué dans la section précédente des algorithmes maximisant d'autres critères, sur les règles d'association par exemple, tels que la confiance (Agrawal et al., 1993), ou bien la corrélation via les mesures de lift et de χ^2 (Brin et al., 1997a).

5.1.3 Graphes étiquetés : *gSpan*

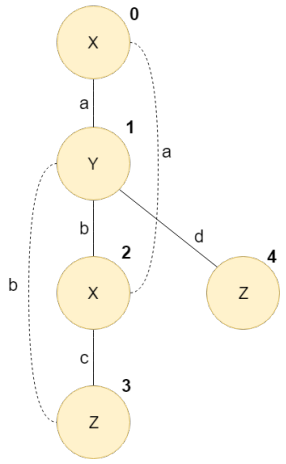
Extraction de motifs dans des graphes étiquetés

D'autres algorithmes d'extraction de motifs prennent des données structurées telles que des graphes voire des graphes étiquetés en entrée. Il se pose immédiatement le problème de la définition du support d'un motif, et de la notion même de motif. Tout d'abord, un motif prend généralement la forme d'un sous-graphe, mais peut également correspondre à un ensemble de nœuds par exemple. Ensuite, doit-on prendre en compte une seule occurrence du motif dans un graphe, ou bien plusieurs ? Un motif doit-il prendre en compte la connexité des graphes qui le supportent ou bien seulement les étiquettes, si présentes ? Le choix de la forme des motifs est toujours d'actualité, et souvent les algorithmes se différencient, d'une part par les techniques employées et les performances, c'est-à-dire leur rapidité d'exécution et leur optimisation de la mémoire, d'autre part par la formalisation des motifs renvoyés, c'est-à-dire la structure et le calcul de support des motifs.

Un second problème se pose aussi : la recherche de sous-graphes fréquents est une tâche bien plus complexe que la recherche de sous-ensembles ou de sous-séquences. Par exemple, si l'on reste dans un cas simple, où la technique de génération de candidats est directement adaptée d'*Apriori*, il n'est pas évident de définir un candidat de taille $k+1$ à partir d'un candidat de taille k. Ainsi les algorithmes proposés par Inokuchi et al., 2000 ; Kuramochi et Karypis, 2001 ; Vanetik et al., 2002 font un choix différent : un candidat y est respectivement étendu d'un nœud, d'un arc ou d'un sous-graphe. Dans ce cas simple, le choix du parcours des candidats n'est déjà pas évident, mais de plus, ces algorithmes résultent en un très grand coût sur leurs performances lorsque la taille des motifs va en grandissant.

L'algorithme *gSpan*

gSpan (Yan & Han, 2002) propose de traiter le problème de la structure en graphe par le biais d'une représentation plus simple. Cet algorithme est populaire dans la communauté de l'extraction de motifs fréquents pour son efficacité et ses concepts algorithmiques.



arc	code DFS
0	(0,1,X,a,Y)
1	(1,2,Y,b,X)
2	(2,0,X,a,X)
3	(2,3,X,c,Z)
4	(3,1,Z,b,Y)
5	(1,4,Y,d,Z)

FIGURE 5.1 – Exemple de graphe et sa représentation sous forme de code DFS exploité par l'algorithme *gSpan* (YAN & HAN, 2002), repris depuis l'article.

Il prend des graphes étiquetés en entrée et renvoie un ensemble de sous-graphes fréquents ainsi que des règles d'association.

Contrairement à *Apriori*, il n'utilise pas la génération de candidats ni la méthode de recherche en largeur. Il traduit d'abord les graphes en une séquence d'étiquettes, noté *minDFS* (minimal Depth First Search). *minDFS* est défini par la séquence minimale, suivant un ordre lexicographique (ou de tout ordre linéaire sur les étiquettes donné), des étiquettes obtenues en effectuant un parcours en profondeur d'un graphe. La figure 5.1.3 représente la traduction de tous les arcs d'un graphe étiqueté en code DFS : par exemple *(0, 1, X, a, Y)* correspond au fait de partir du nœud *0* et aller au nœud *1* lors du parcours. *minDFS* est obtenu en choisissant le parcours en profondeur qui minimise la séquence d'étiquettes obtenue selon l'ordre sur les étiquettes. On part d'un des nœuds de plus petite étiquette, au sens de l'ordre sur les étiquettes, puis on va de nœud en nœud en choisissant à chaque fois les étiquettes d'arc et de nœud le plus petit possible, et en suivant une logique de parcours en profondeur.

Les auteurs justifient la validité de l'utilisation des codes *minDFS* par le théorème suivant : si deux graphes sont isomorphes l'un à l'autre, leurs codes *minDFS* sont identiques. La tâche d'exploration de sous-graphes fréquents devient donc une tâche d'exploration de sous-séquences fréquentes.

Représenté dans la figure 5.2, page 125, un arbre dont les étiquettes des nœuds sont des *minDFS* est construit, chaque nœud de l'arbre étant un code DFS d'un graphe plus

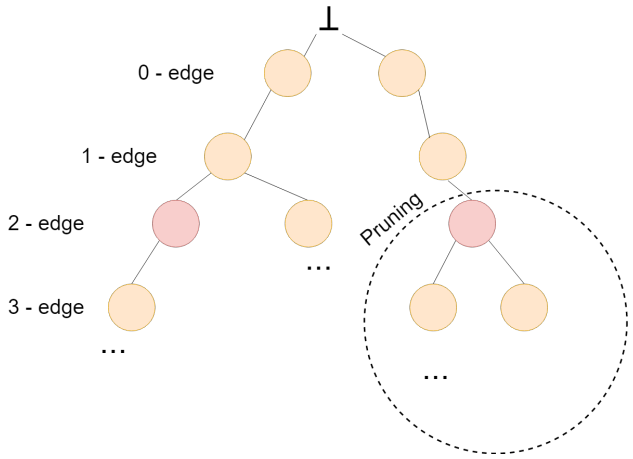


FIGURE 5.2 – Elagage de l'arbre des codes DFS similaires effectué par *gSpan* (YAN & HAN, 2002), repris de l'article.

grand d'un arc par rapport à son parent, comme extension de son parent. Cet arbre de code DFS est le nouvel espace de recherche. Contrairement à *Apriori*, qui étend les candidats tous en même temps, gSpan effectue un parcours en profondeur dans l'arbre dans candidats. Cela résulte en la formation d'un motif le plus grand possible, par extension puis calcul de support successifs. Grâce à ce choix, une fois arrivé sur un nœud correspondant à un code *minDFS* de support inférieur au seuil fixé, tous les descendants sont élagués ainsi que ceux des nœuds de même code présents ailleurs dans l'arbre. Cet élagage est également représenté dans la figure 5.2, page 125, avec le dernier candidat encore fréquent, du parcours en profondeur courant, qui est coloré en rouge. Cette technique permet de grandement réduire l'espace de recherche.

En conclusion, *gSpan* combine un certain nombre de qualités maximisant l'efficacité lors de l'extraction de motifs fréquents dans des graphes étiquetés. Certains algorithmes s'en sont ainsi inspirés, y compris pour des formalismes sémantiquement plus riches que les graphes étiquetés, comme c'est le cas dans l'algorithme *DMGM-GSM* présenté ensuite.

5.1.4 Graphes étiquetés avec taxonomie : *DMGM-GSM*

Les algorithmes d'extraction de motifs fréquents prenant en entrée les données de structure la plus proche de celle des graphes conceptuels sont sans doute ceux fonctionnant sur des graphes étiquetés avec taxonomie (INOKUCHI, 2004 ; CAKMAK & OZSOYOGLU, 2008 ; PETERMANN et al., 2017). Ils se différencient notamment sur les choix algorithmiques mais également sur les critères d'intérêt à maximiser : selon l'algorithme, au lieu de chercher des motifs fréquents, ce sont plutôt des motifs fréquents classés par degré

d'intérêt ou respectant un seuil minimal qui sont renvoyés.

Dans ces algorithmes, cette relation d'ordre sur les étiquettes permet de construire des motifs généralisés, où la correspondance exacte avec une partie d'un graphe de la base d'entrée n'est plus nécessaire pour incrémenter le support. Il suffit en effet que l'étiquette du motif soit égale ou bien plus générale que l'étiquette du graphe sur laquelle elle est projetée, de façon similaire à la relation d'homomorphisme rappelée dans la section 1.2.5, page 33, pour le cas des graphes étiquetés avec taxonomie.

Un des risques est que certains des motifs renvoyés soient trop généraux, par exemple constitués seulement de l'étiquette la plus générale, et donc très peu informatifs. L'utilisation de critères d'intérêt prend alors toute son importance, en plus de généralement d'accroître la qualité des résultats. Par exemple, Inokuchi, 2004 propose de se limiter aux motifs qui ne sont pas sur-généralisés. Une première idée est d'abord de renvoyer tous les motifs fréquents, en permettant la généralisation des étiquettes, puis de retirer les motifs sur-généralisés. Cependant, cela conduit à un nombre trop important de candidats dans la première étape. Inokuchi, 2004 propose alors de calculer un support dit pondéré, qui correspond au nombre total d'occurrences d'un motif, comptant ainsi plusieurs fois s'il est présent plusieurs fois dans un même graphe. Ensuite, ce support est utilisé pour détecter les motifs sur-généralisés : tout motif A plus général qu'un motif B et de support pondéré égal est retiré. Inokuchi, 2004 avance que c'est à cette condition qu'aucun motif d'intérêt n'est perdu lors de l'élagage.

Ainsi, l'utilisation d'un tel critère permet de gagner en qualité et en efficacité lors de l'extraction de motifs fréquents. *DMGM-GSM* (Petermann et al., 2017) se démarque d'une part par la prise en compte de plusieurs taxonomies sur les étiquettes ; d'autre part par une première extraction de motifs qui se limite à ne prendre en compte que la structure, suivie d'une spécialisation de ces motifs structurels identifiés pour obtenir les motifs renvoyés ; et enfin, par l'utilisation comme critère de qualité d'un degré de significativité statistique des motifs. Il se base sur *gSpan* (Yan & Han, 2002) ainsi que sur Inokuchi, 2004.

Tout d'abord, l'extraction des motifs structurels consiste à appliquer *gSpan* sur les graphes avec les étiquettes généralisées au maximum. Pour ce faire, et pour prendre en compte les taxonomies, chaque étiquette est remplacée par un chemin vers l'étiquette la plus générale possible. Ainsi, en reprenant l'exemple des attaques informatiques et la hiérarchie d'étiquettes représentées dans la figure 1.3, page 25, une étiquette *FichierImportant* serait remplaçée par l'étiquette *FichierImportant_Fichier_ÉlémentSystème_*$\top$. Dans cette étape, seule l'étiquette la plus générale, $\top$, est prise en compte.

Puis *DMGM-GSM* spécialise les étiquettes une à une pour constituer d'autres candidats, et stoppe la spécialisation dès que les candidats voient leur support passer sous le

seuil fixé. L'algorithme dans son implémentation actuelle renvoie tous les motifs ainsi trouvés au sein d'un treillis dont la relation d'ordre est la généralisation.

Enfin, dans l'article, les algorithmes sont discriminés par le calcul d'un degré de significativité statistique inspiré du modèle de Chung et Lu, 2002 et de son adaptation au cas des graphes étiquetés par Micale et al., 2018. Le calcul n'est pas explicité par l'article, mais le principe est de générer des bases aléatoires possédant des caractéristiques similaires aux bases sur lesquelles par l'algorithme d'extraction est lancé. Ces caractéristiques sont à fixer, mais correspondent notamment pour Micale et al., 2018 à la distribution du degré des nœuds au sein d'un graphe. On peut supposer que dans le cas de *DMGM-GSM* et donc des graphes étiquetés avec taxonomie, cette distribution peut prendre en compte les étiquettes voire leur degré de généralisation (par exemple calculé comme la distance à la racine). Ensuite, ces bases ainsi générées sont également fouillées par l'algorithme d'extraction. Plus les motifs construits sont différents des motifs obtenus dans la base d'entrée, plus ces derniers sont de qualité. Les motifs les plus intéressants sont alors définis comme les plus *étonnants* statistiquement au regard des caractéristiques de base choisies. Petermann et al., 2017 proposent un classement des motifs en fonction de ce critère.

Le choix de *DMGM-GSM* est d'utiliser le critère d'intérêt pour classer les motifs plutôt que les élaguer. C'est un choix effectué par des algorithmes dans l'état de l'art (Stern et al., 2006 ; Cakmak & Ozsoyoglu, 2008 ; Gwadera & Crestani, 2010 ; Yin et al., 2011) pour renvoyer un résultat plus complet selon différents critères (Chen, 2007 ; Dashti et al., 2010 ; De Runz et al., 2021).

5.2 Algorithme proposé pour les graphes conceptuels

Nous proposons *cgSpan*, basé sur *DMGM-GSM* (Petermann et al., 2017) présenté dans la section précédente, comme un algorithme d'extraction de motifs fréquents depuis une base de connaissances sous forme de graphes conceptuels. Nous considérons les graphes conceptuels comme des graphes étiquetés avec taxonomie avec plus de sortes de connaissances représentées. *cgSpan* prend en compte les étapes consécutives d'enrichissement des graphes étiquetés avec taxonomie aux graphes conceptuels. L'objectif est de montrer que l'utilisation de ces spécificités permet d'obtenir un algorithme plus efficace en termes d'espace mémoire, de vitesse et de qualité des résultats par rapport aux algorithmes existants sur les graphes étiquetés avec taxonomie.

Nous proposons d'exploiter trois différences avec le modèle des graphes étiquetés avec taxonomie : les relations à arité fixe, les signatures et les règles d'inférence. La première différence correspond à la biparticité des graphes conceptuels, la seconde aux

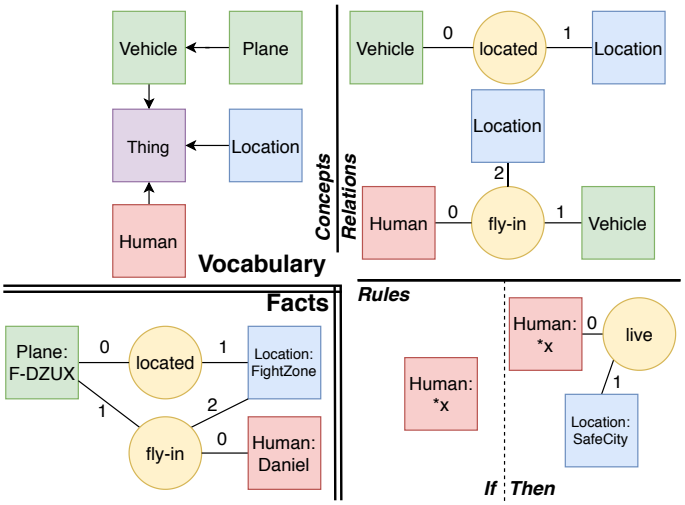


FIGURE 5.3 – Exemple d'un graphe conceptuel avec son vocabulaire : un graphe conceptuel dans la partie inférieure gauche, une hiérarchie de concepts dans la partie supérieure gauche, deux signatures dans la partie supérieure droite et une règle d'inférence dans la partie inférieure droite.

contraintes sur les étiquettes des nœuds concept connectés aux nœuds relation et la troisième aux mécanismes de déduction.

La section 5.2.1, page 129 décrit l'algorithme proposé. Les modules prenant en compte les trois spécificités des graphes conceptuels considérées sont ensuite présentées dans les sections 5.2.2 à 5.2.4.

Dans toute cette section, nous considérons à titre d'exemple des graphes conceptuels construits sur les connaissances ontologiques représentées sur la figure 5.3, page 128, utilisées pour représenter des mouvements de véhicules. Les relations sont au nombre de 3, avec *live* et *located* d'arité 2, et *fly-in* d'arité 3. Les signatures de ces deux dernières sont renseignées en haut à droite : elles indiquent respectivement la localisation d'un véhicule d'une part, et le conducteur et la localisation d'un véhicule d'autre part. La règle dans l'ontologie représente que tous les humains vivent dans *SafeCity*. Enfin, le graphe conceptuel représente un avion, *F-DZUX*, piloté par l'humain *Daniel* et situé dans *FightZone*.

La hiérarchie T_R n'est pas représentée : les relations d'arité 2 et 3 sont respectivement des spécialisations de T_2 et T_3.

5.2.1 Vue d'ensemble

cgSpan est composé de trois modules, chacun ayant pour objectif la prise en compte d'une spécificité des graphes conceptuels par rapport aux graphes étiquetés avec taxonomie : les règles d'arité fixe, les signatures et les règles d'inférence. Ils peuvent être combinés pour exploiter simultanément toutes les caractéristiques des graphes conceptuels, comme illustré dans la section 5.3, page 134. Ils sont décrits tour à tour dans les sections suivantes et la présente section décrit leurs points communs.

La qualité évoquée tout au long de cette section, et des suivantes, correspond à un critère de non-redondance des motifs extraits avec les connaissances ontologiques et implicites. Ces connaissances sont les trois caractéristiques prises en compte par *cgSpan*. Ce critère est détaillé en section 5.3.2, page 135.

La partie essentielle de *cgSpan* est une étape de prétraitement qui traduit chaque graphe conceptuel de la base de données d'entrée en graphes étiquetés avec taxonomie. Le principe de base est similaire à celui de *DMGM-GSM* (voir section 5.1.4, page 125) : les étiquettes sont construites en concaténant tous les types de concept sur le chemin taxonomique entre le type spécifié dans chaque nœud et le type qui le généralise le plus, en les séparant par un '_'. Si un marqueur individuel est présent, il est renseigné en tant que type plus spécifique dans le chemin taxonomique. Par exemple, "Plane :F-DZUX" de la figure 5.3 est remplacé par "⊤_Vehicle_Plane_F-DZUX".

Ce principe général est modifié au moyen des trois modules pour prendre en compte les spécificités des graphes conceptuels. Le premier concatène les étiquettes d'une relation et les nœuds qui lui sont connexes ; le second applique les règles d'inférence correspondant à un nœud ; enfin le troisième limite les étiquettes afin qu'elles se conforment aux signatures de relation. Ces modifications visent à accroître la qualité des résultats et l'efficacité de l'algorithme, notamment en diminuant l'espace de recherche.

Ensuite, l'étape d'extraction de motifs est effectuée en appliquant *DMGM-GSM*, modifié par le module de règles d'inférence par l'application de règles, qui évite ainsi d'explorer certains motifs pour une plus grande qualité et efficacité.

Enfin, le post-traitement traduit les motifs fréquents obtenus dans le formalisme des graphes conceptuels. Le module de signature comprend une étape supplémentaire d'élagage permettant de ne garder que les motifs conformes à ces signatures. En effet, malgré l'étape de conformité effectuée initialement, l'étape d'extraction de motifs conduit en règle générale à l'introduction de motifs non conformes, notamment lors de l'application de règles-λ. Le but de cette étape est d'augmenter l'efficacité de l'algorithme et la pertinence des motifs extraits.

5.2.2 Exploitation de l'arité des relations

La première spécificité des graphes conceptuels que nous proposons d'exploiter est l'arité des nœuds relation, spécifiée par la hiérarchie ou les signatures : tout nœud relation est nécessairement connexe à un nombre connu de nœuds concept. Les types de nœuds concept sont également contraints par les signatures, mais sont gérés dans le module suivant.

Ainsi, dans l'exemple illustratif en figure 5.3, page 128, les signatures représentées permettent d'avoir l'arité des relations *located* et $fly-in$, respectivement d'arité 2 et 3.

Le principe général de ce module est d'assurer la cohérence des connaissances renvoyées par les motifs en assurant que tout nœud relation est nécessairement accompagné de tout son voisinage. D'autre part, lors de l'étape d'incrémentation des candidats fréquents, nous ne voulons pas que les différentes candidats constitués de voisinages partiels de relation soient ajoutés.

Cette double qualité prend forme dans ce module par une structuration des graphes conceptuels autour des relations. Cela est d'autant plus cohérent avec l'idée que les graphes conceptuels sont une représentation graphique visuelle de formules de la logique des prédicats du premier ordre.

Nous définissons ainsi une *brique élémentaire* comme un nœud relation et son voisinage : les nœuds concept qui lui sont connexes. Le découpage des graphes conceptuels en briques élémentaires plutôt qu'en nœuds permet d'éviter une perte de sens des motifs construits, en évitant que des relations n'aient qu'une partie de leur voisinage dans ces motifs, et d'accroître l'efficacité de l'extraction de motifs. La figure 5.4, page 131 illustre comment un graphe conceptuel peut être traduit en un graphe étiqueté avec taxonomie dont les nœuds sont les briques élémentaires du graphe conceptuel. Les étiquettes de chemins taxonomiques détaillées dessous partent des types maximaux et spécialisent toutes les étiquettes de nœuds en même temps. Dans l'exemple représenté, "T3" est un type de relation plus générale que "fly-in".

Une brique est associée à un seul nœud du graphe étiqueté avec taxonomie. Une difficulté reste à écarter : les étiquettes pour *DMGM-GSM* sont des chemins taxonomiques, et la traduction d'un ensemble de chemin taxonomiques pour chaque élément du voisinage, en un seul chemin taxonomique de brique élémentaire n'est pas évidente. Il peut être choisi de modifier *DMGM-GSM* pour prendre en compte des étiquettes plus complexes comme conjonction de chemin taxonomique par exemple. Nous choisissons de définir un chemin taxonomique concaténé, où chaque étape du chemin est plus général que le précédent. Ainsi, figure 5.4, page 131, $fly-in(Human, Plane, Location)$ est traité en généralisant chacune des sous-étiquettes qui la compose. Il pourrait être fait le choix de ne généraliser qu'une étiquette à la fois, mais cela entraîne bien plus de motifs renvoyés

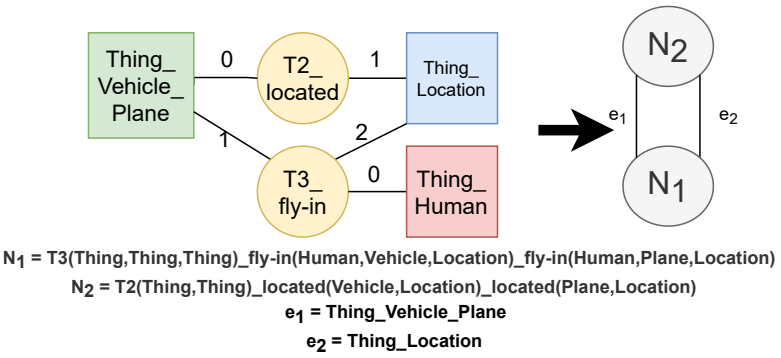


FIGURE 5.4 – Traduction de graphe conceptuel en un graphe étiqueté avec taxonomie dont les nœuds sont des briques élémentaires

et nous voulons ici d'abord privilégier l'efficacité.

Une étiquette de brique élémentaire est définie comme suit. Il s'agit de la concaténation des étiquettes du nœud relation et de chacun de ses nœuds concept associés, dans l'ordre spécifié dans les graphes conceptuels par les étiquettes des arêtes. Ainsi, pour la relation $fly-in$, on a dans l'ordre l'étiquette de l'humain, de l'avion et du lieu. Les chemins taxonomiques sont inclus en généralisant successivement toutes les sous-étiquettes en même temps, jusqu'à ce que toutes atteignent le type le plus général de leur chemin taxonomique respectif. Tel que renseigné dans la transcription des étiquettes de briques élémentaires représentées en bas de la figure 5.4, la généralisation consiste ainsi à remplacer *Location* par $\top$, *Plane* par *Vehicle*, *Human* par $\top$ et $fly-in$ par T_3.

Ensuite, les arêtes sont construites entre les briques qui partagent un nœud concept commun, comme illustré dans la partie droite de la figure 5.4. Les étiquettes d'arêtes sont les chemins taxonomiques des nœuds concept communs aux briques élémentaires. La figure 5.4 montre par exemple la transformation d'une brique élémentaire dans un graphe conceptuel avec "A plane is flown by a human in a place" en N_1.

5.2.3 Exploitation des signatures

La deuxième spécificité des graphes conceptuels considérée dans *cgSpan* est celle des signatures. Elles définissent pour chaque type de relation une restriction sur les types de concept : pour chaque nœud relation, elles spécifient un type maximal pour chaque nœud concept qui lui est connexe. Nous proposons d'exploiter cette information dans les étapes de pré-traitement et de post-traitement, respectivement pour restreindre la généralisation des étiquettes des nœuds concept et pour élaguer les motifs afin d'éviter la redondance avec les signatures. L'objectif est d'éviter de n'identifier comme motifs fréquents que les signatures qui constitueraient un résultat sans intérêt. Ce module, de

manière similaire à Inokuchi, 2004, vise ainsi à éviter des motifs trop généralisés.

A titre d'exemple, nous considérons dans cette section la relation "fly-in" de signature (*Human*, *Vehicle*, *Location*) comme illustré dans la figure 5.4.

Dans l'étape de prétraitement, nous proposons d'élaguer le chemin de la taxonomie, non pas en le construisant jusqu'au type le plus général, mais jusqu'au niveau indiqué dans la signature. Pour chaque nœud relation dans un graphe conceptuel, les nœuds concept connectés à cette relation voient leurs étiquettes comparées à la signature correspondante. Chaque comparaison est ensuite suivie d'une coupe du chemin de la taxonomie jusqu'au type correspondant dans la signature. Par exemple, la brique élémentaire "fly-in(⊤_Human,⊤_Vehicle_Plane,⊤_Location)" est remplacée par "fly-in-(Human,Vehicle_Plane, Location)" dans la figure 5.4, page 131.

Dans l'étape de post-traitement, deux types de motifs sont affectés. Premièrement, nous proposons de supprimer les motifs qui correspondent à un ensemble de signatures connectées entre elles : de tels motifs n'apportent rien ou presque par rapport aux connaissances du vocabulaire. Deuxièmement, nous proposons de supprimer les sous-parties des motifs qui correspondent à un ensemble de signatures connectées entre elles : nous appelons cette opération la réduction de motif. Chaque signature ainsi supprimée est remplaçée par un nœud constitué d'une étiquette concise qui référence la signature remplacée.

Par exemple, "fly-in(Human,Vehicle,Location)" est élaguée, ce qui correspond à la première opération de suppression des motifs redondants avec les signatures. "fly-in(Human, Vehicle_Plane, Location)" quant à lui ne l'est pas, puisque "Vehicle_Plane" est plus précis que "Vehicle" spécifié dans la signature comme illustré dans la figure 5.5, page 133 : il correspond seulement partiellement à une signature, et cette partie est remplacée par une référence à la signature correspondante.

5.2.4 Exploitation des règles d'inférence

La troisième spécificité prise en compte par *cgSpan* est l'existence des règles d'inférence, en distinguant celles qui entraînent une spécialisation et celles qui entraînent une extension. Lors de la construction des candidats, si un candidat correspond à une prémisse de règle, il est alors remplacé par sa conclusion.

Une règle entraînant une spécialisation spécifie que lorsqu'un graphe conceptuel inclut la prémisse de la règle, certains nœuds spécifiés dans la conclusion de la règle peuvent être spécialisés. Ainsi si la prémisse recouvre une sous-partie du candidat courant, elle lui est appliquée. Lors du prétraitement, après le remplacement par des chemins de taxonomie dans les graphes conceptuels, nous proposons d'étendre ce chemin pour chaque nœud correspondant à l'hypothèse d'une telle règle au type de la conclu-

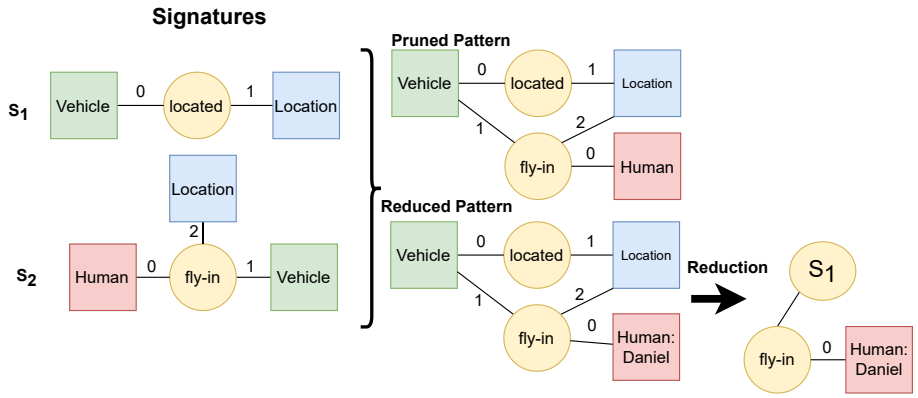


FIGURE 5.5 – Élagage des motifs durant l'étape de post-traitement prenant en compte les signatures (voir section 5.2.3, page 131). Le motif en haut à droite est élagué car il s'agit d'une simple agrégation de signatures. Le motif en bas à droite est réduit en remplaçant la partie redondante de ce motif par une référence à la signature.

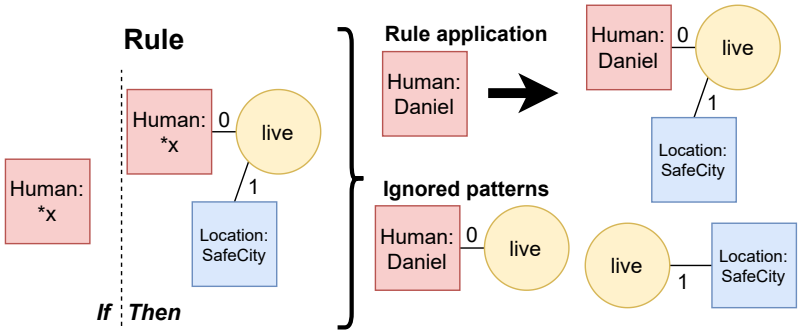


FIGURE 5.6 – Exemple d'application d'une règle et exemples de motifs non explorés par cette technique

sion.

Au contraire, une règle entraînant une extension ajoute des connaissances à des graphes conceptuels. Durant l'étape d'exploration de motifs, nous proposons d'étendre tous les motifs correspondant à la conclusion de la règle. Au cours du processus, la prémisse de la règle et les motifs intermédiaires sont ignorés, c'est-à-dire qu'ils ne se retrouveront pas dans le résultat, tel que représenté figure 5.6. Elle illustre l'application de la règle de la figure 5.3, page 128. Tous les motifs intermédiaires, de l'hypothèse à la conclusion, ne sont pas explorés lors de l'utilisation de telles règles puisque l'extension se fait en une seule étape.

Les règles d'inférence peuvent être appliquées plusieurs fois, ce qui permet d'étendre un graphe conceptuel correspondant à chaque application successive, mais nous limitons

leur utilisation à une seule.

5.3 Étude expérimentale

Cette section décrit les expériences que nous avons menées, en présentant les données synthétiques et les critères de qualité considérés et en discutant les résultats obtenus. L'objectif de *cgSpan* est de maximiser l'efficacité en terme d'espace mémoire et temps de calcul, et également la pertinence des résultats au moyen de la maximisation d'un critère de non-redondance.

La méthode de génération de données utilisées est d'abord présentée dans la section 5.3.1 qui suit, puis les critères utilisés ainsi que leur calcul sont explicités en section 5.3.2, page 135, et enfin les résultats expérimentaux en section 5.3.3, page 136.

Le chapitre 6 présente par ailleurs les résultats de *cgSpan* sur un cas d'application industriel.

5.3.1 Génération de données

Nous utilisons *CG2A* (Faci et al., 2021a), présenté dans le chapitre 4, page 97 pour générer des bases de graphes conceptuels. Ces dernières comprennent des résultats attendus, les graphes conceptuels-γ utilisés en entrée de *CG2A* ainsi que toute combinaison possible de ces graphes respectant les contraintes numériques et ontologiques.

Nous utilisons deux jeux de données dans les résultats expérimentaux, notés D_1 et D_2. D_1 est généré par *CG2A-FullAuto* avec $nbGC = 1000$, $minSize = 25$, et un $maxSpe = 2$. La profondeur des hiérarchies est fixée à 4 pour les concepts et 3 pour les relations, le nombre maximum de spécialisations par type est 3 et le nombre de marqueurs individuels par type de concept est 2. 10 graphes conceptuels-γ sont générés et ont une taille d'au moins 7. Enfin, chaque graphe conceptuel-γ a une variable de concept, de relation et de marqueur respectivement, dont le nombre de valeurs respectif est 3 et le nombre de spécialisations maximal est 3. 5 règles d'inférence ont ensuite été générées aléatoirement à partir du vocabulaire généré.

Pour D_2, CG2A est configuré de manière similaire, à une différence : afin de disposer de résultats attendus d'une forme différente et ouvrir la discussion, D_2 est généré en suivant une distribution sur la taille des graphes conceptuels, renseignée figure 5.7, page 137.

D_1 et D_2 sont constitués de 1000 graphes d'environ 30 nœuds chacun. Les hiérarchies contiennent 50 types de concept et 20 types de relation.

D_1 permet de définir les résultats attendus. Tout d'abord, nous nous attendons à ce que l'ensemble des graphes conceptuels prédéfinis utilisés comme entrée pour générer

ces ensembles de données soient présents dans les motifs fréquents. Ensuite, nous nous attendons à ce que certaines contraintes simples vérifiées par D_1 soient également vérifiées par l'ensemble des motifs fréquents. Nous nous attendons également à ce que la distribution sur les étiquettes dans D_2 soit suivie par les motifs.

5.3.2 Critères

Dans cette section sont définis les critères de qualité utilisés pour évaluer *cgSpan*.

Rappel :

Ce critère, noté Rap. dans les tableaux de résultats, est défini comme la proportion de motifs attendus présents dans les motifs renvoyés. Il doit être égal à 1 pour garantir que l'algorithme est fonctionnel.

$$rappel = \frac{\{motifs\ attendus\} \cap \{motifs\ renvoyés\}}{\{motifs\ attendus\}}$$

.

Précision :

Ce critère, noté Pre., est défini comme la proportion de motifs renvoyés qui sont présents dans les motifs attendus. Elle doit être proche de 1 car l'algorithme peut renvoyer des motifs inattendus intéressants. Les résultats doivent ensuite être examinés qualitativement.

$$precision = \frac{\{motifs\ attendus\} \cap \{motifs\ renvoyés\}}{\{motifs\ renvoyés\}}$$

.

Redondance :

Ce critère, noté Red., correspond à la proportion de motifs élagués par rapport à l'ensemble des motifs, renvoyés ou élagués. Il est calculé lors de la suppression des motifs par le module des signatures et par l'évitement des motifs lors de l'application du module de règles d'inférence. Plus il est élevé, plus la redondance a été supprimée, meilleur est le résultat.

$$redondance = \frac{\{motifs\ élagués\}}{\{motifs\ renvoyés\} \cup \{motifs\ élagués\}}$$

.

Efficacité temporelle :

Ce critère, noté Eff., compare le temps d'exécution de l'algorithme testé t_{algo} au temps d'exécution de *DMGM-GSM* $t_{DMGM-GSM}$ pris comme référence. Ce critère est à minimiser.

$$t - efficacité = \frac{t_{cgSpan}}{t_{DMGM-GSM}}$$

5.3.3 Résultats expérimentaux

Nous traitons D_1 et D_2 traduits en graphes étiquetés avec *DMGM-GSM*[1] et les traitons avec quatre variantes de *cgSpan*, en considérant chaque module, décrit en sections 5.2.2 à 5.2.4, et *cgSpan* complet qui inclut les trois modules.

Le tableau 5.1, page 137 donne les résultats obtenus pour D_1. La flèche à côté de chaque critère indique s'il faut le maximiser (↑) ou le minimiser (↓). Tous les algorithmes identifient tous les motifs ou combinaisons de motifs attendus et atteignent donc l'objectif fonctionnel, en obtennant un rappel égal à 100%. En ce qui concerne la précision, toutes les variantes ne prenant pas en compte les briques élémentaires renvoient des briques élémentaires partielles, réduisant leurs performances pour ce critère. L'utilisation des signatures permet d'élaguer certains de ces motifs incomplets puisque la plupart d'entre eux contiennent des nœuds concept de type plus général ou incompatible avec ce qui est spécifié dans la signature. On peut remarquer qu'une augmentation de la redondance est corrélée à une augmentation de la précision. Cela peut être interprété comme "Les modèles qui sont élagués sont pour la plupart des modèles non attendus". Enfin, l'utilisation de briques élémentaires semble augmenter l'efficacité temporelle de manière significative. En effet, on observe un gain de près de 20% en temps alors que le gain avec l'utilisation de règles est inférieur à 10%.

La figure 5.7, page 137 présente les résultats obtenus par *cgSpan* sur D_2, montrant le nombre d'occurrences des motifs attendus et identifiés en fonction de leur taille. La distribution de fréquence sur le jeu de données d'entrée a été utilisée pour générer D_2, nous utilisons donc les deux distributions pour analyser les résultats. Il faut noter que lors du comptage du support des motifs, seuls les motifs maximaux, c'est-à-dire les motifs non inclus dans un autre motif, contribuent à cette illustration : tous les motifs comptés de taille 3 ne sont pas présents dans les motifs comptés de taille supérieure. On peut observer une corrélation globale entre le jeu de données d'entrée et les motifs fréquents concernant ces distributions. Le fait qu'il n'y ait pas de motif fréquent identifié de taille 1 ou 2 s'explique par la restriction du premier module de *cgSpan* qui ne permet pas de brique élémentaire constituée d'un voisinage partiel alors que dans D_2 ces graphes conceptuels

1. Nous utilisons l'implémentation de l'algorithme fourni par les auteurs sur la page Petermann, s. d.

TABLE 5.1 – Comparaison de *cgSpan* et ses variantes sur les données D_1

Test	Rap. (%) ↑	Pre. (%) ↑	Red. (%) ↑	Eff. (%) ↓
DMGM-GSM	100	57	0	100
cgSpan avec briques	100	83	0	83
cgSpan avec signatures	100	75	38	97
cgSpan avec règles	100	63	20	93
cgSpan complet	100	85	46	79

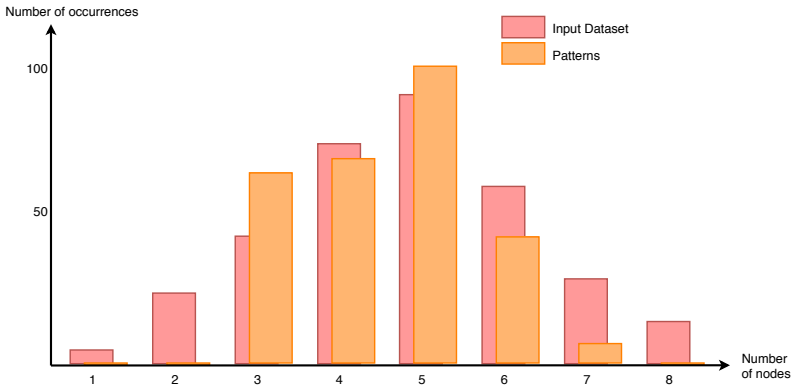


FIGURE 5.7 – Distributions de fréquence sur D_2 et sur l'ensemble des motifs fréquents obtenus avec *cgSpan*

correspondent à des nœuds concept isolés.

5.4 Bilan

cgSpan est un algorithme permettant d'extraire des motifs fréquents d'une base de données de graphes conceptuels. Les particularités des graphes conceptuels sont exploitées pour produire un algorithme fonctionnel, et la qualité des résultats est améliorée par rapport aux critères définis. Nous utilisons la différence entre les concepts et les relations et les règles d'inférence pour accélérer le processus et nous utilisons les signatures et d'autres éléments pour élaguer les motifs moins pertinents.

Les perspectives de ces travaux incluent l'extension de l'algorithme *cgSpan* proposé, en considérant des variantes encore plus riches de graphes conceptuels, en particulier des graphes conceptuels imbriqués (CHEIN & MUGNIER, 2008). Une autre direction pourra se concentrer sur la définition de critères de qualité. Par exemple il pourrait être intéressant d'exploiter une significativité statistique basée sur le modèle de CHUNG et LU, 2002, adap-

tée au cas des graphes étiquetés avec taxonomie par Micale et al., 2018. Cela permettrait d'améliorer à la fois la qualité des résultats et la pertinence de la validation.

Une piste intéressante est également la mise en place d'une implémentation des différents critères permettant de qualifier un motif proposés par l'état de l'art, ceci afin de caractériser totalement un motif, et obtenir des résultat plus complets.

Enfin, tester *cgSpan* sur un cas appliqué permettrait de confronter ses apports algorithmiques à des besoins métiers, et de compléter sa validation sur des données réelles. C'est ce qui est effectué dans la section suivante avec l'extraction de motifs dans des trajectoires sous forme de graphes conceptuels.

Chapitre 6

Application : caractérisation de données de trajectoires

Ce chapitre présente les travaux applicatifs menés dans un cadre industriel, avec l'entreprise *THALES*, en lien avec le projet européen *H2020 Camelot*.

Leur objectif global est la caractérisation de données de trajectoires pour la sécurité maritime : des zones côtières sont surveillées par différents capteurs qui détectent des véhicules. Les trajectoires des véhicules détectés sont classées selon différents scénarios, chacun correspondant à un jeu de données différent, étudié pour obtenir un profil de trajectoire. Ce profil est caractéristique d'un type de véhicule et d'un contexte.

Le but visé est de caractériser ce profil de trajectoire à partir d'un ensemble de données de trajectoires. Les données sont formalisées sous forme de graphes conceptuels. Nous proposons donc d'utiliser *cgSpan*, présenté dans le chapitre précédent, pour extraire de ces données des motifs fréquents qui constituent de tels profils de trajectoire. Nous nous attendons à ce qu'il y ait des motifs communs à toutes les trajectoires, c'est-à-dire caractérisant le concept de trajectoire, ainsi que des motifs spécifiques à un profil de trajectoire, c'est-à-dire ceux caractérisant un profil en particulier : ce sont ces derniers qui nous intéressent pour répondre au besoin métier.

Nous présentons en section 6.1 le plan d'expérience établi pour répondre à la problématique présentée, puis nous en détaillons les résultats en section 6.2, page 144.

6.1 Plan d'expérience

Cette section présente les conditions générales des expériences menées que nous détaillons dans les sous-sections qui suivent.

Nous disposons de 9 scénarios, et donc 9 jeux de données, dont les 3 les plus repré-

sentatifs sont décrits dans la section 6.1.2, page 142. Les 6 autres sont des variantes, dont les résultats sont pris en compte en section 6.2.2, page 147 mais ne nous ont pas fourni des résultats sensiblement différents pour permettre de les aborder dans les résultats de la section 6.2.3, page 149.

Les données sont formalisées par des graphes conceptuels, dont le vocabulaire n'est que partiellement disponible : il ne fournit pas les signatures utilisées et ne contient pas de règle d'inférence. Nous ne pouvons donc valoriser *cgSpan* que par son premier module de briques élémentaires, introduit dans la section 5.2.2, page 130. Nous proposons cependant d'enrichir le vocabulaire de signatures, suite à une analyse préliminaire des données, en section 6.2.1, page 145. Cet ajout de signatures permet d'également tester *cgSpan* avec le module de signature, décrit dans la section 5.2.3, page 131.

Nous effectuons ces expériences en comparant les résultats de *DMGM-GSM* (Petermann et al., 2017) avec trois variantes de *cgSpan* : *cgSpan-n*, *cgSpan-s* et *cgSpan-ns* utilisant respectivement le module de briques élémentaires, le module de signatures et la combinaison de ces deux modules. La qualité est évaluée en fonction de la cohérence des motifs construits, et leur pertinence à représenter les spécificités du scénario associé. Pour *cgSpan-n*, cette qualité est due à la garantie que les motifs construits comprennent uniquement des nœuds relation dont le voisinage est complet : il y a ainsi un gain en cohérence des motifs. Pour *cgSpan-s*, cette qualité est assurée par le respect des signatures par les motifs et la suppression des motifs redondants avec les signatures : il en résulte un gain en cohérence et en pertinence des motifs. Enfin, il est attendu que *cgSpan-ns* réunisse ces deux qualités.

Dans ce qui suit, la section 6.1.1 présente les données de trajectoires utilisées. Ensuite, les différents jeux de données, les scénarios, sont abordés en section 6.1.2, page 142. Enfin la section 6.1.3, page 143 décrit les critères utilisés dans la partie résultats pour évaluer et discuter des motifs renvoyés.

6.1.1 Données

La figure 6.3, page 143 représente un exemple de zone observée où 2 véhicules se déplacent. Le point de départ des véhicules est en rouge et leur arrivée en vert. Les 2 trajectoires sont représentées en pointillés. Les rectangles représentent le champ d'action de deux capteurs de drones de surveillance. Dans les autres scénarios, il existe également des segments circulaires représentant des zones surveillées par des radars fixes.

Les données fournies sont les connaissances ontologiques à partir desquelles sont construits des graphes conceptuels de trajectoires, successivement présentés dans cette section.

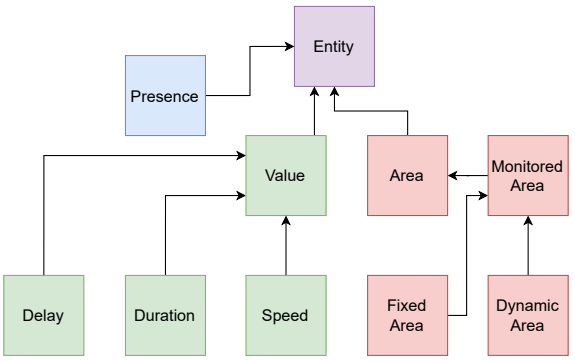


FIGURE 6.1 – Ensemble T_C associé à des trajectoires réduit aux concepts présents dans les jeux de données traités

Connaissances ontologiques

Le vocabulaire comprend une hiérarchie de concepts composée de 32 types de profondeur de 2 ainsi que 27 relations d'ordre.

La hiérarchie, limitée aux concepts effectivement présents dans les données que nous avons traitées, est représentée figure 6.1.

Le type *FixedArea* correspond à une zone surveillée par un radar fixe tandis que le type *DynamicArea* correspond à une zone surveillée par un drone.

Les relations présentes sont *with-speed*, *in*, *precedes*, ainsi que *relation*. *relation* est le nom de deux relations homonymes d'arité respective 2 et 3, qui généralisent respectivement *with-speed* d'une part, et *in* et *precedes* d'autre part. On établit, pour être conforme au formalisme des graphes conceptuels, qu'il existe deux relations homonymes d'arité différentes.

Connaissances factuelles

Les connaissances factuelles représentent les trajectoires sous forme de graphe conceptuel. Chaque graphe est composé d'une unité d'information de trajectoire ou ensemble d'unités connectées par un nœud relation de type *precedes*. Chacune de ces unités d'information comprend un nœud concept de type *Presence* représentant la détection d'un véhicule. Le reste de l'unité fournit la vitesse du véhicule, la zone dans laquelle il a été détecté et la durée de sa présence détectée. Enfin, la relation *precedes* qui connecte les unités entre elle renseigne sur le temps écoulé entre les deux détections.

Les unités de mesure des valeurs de temps et de vitesse ne sont pas fournies, nous parlerons donc ici en terme de valeurs numériques sans unité de mesure. Les valeurs dans des graphes conceptuels ont été introduites dans la section 1.2, page 21. Nous détaillons

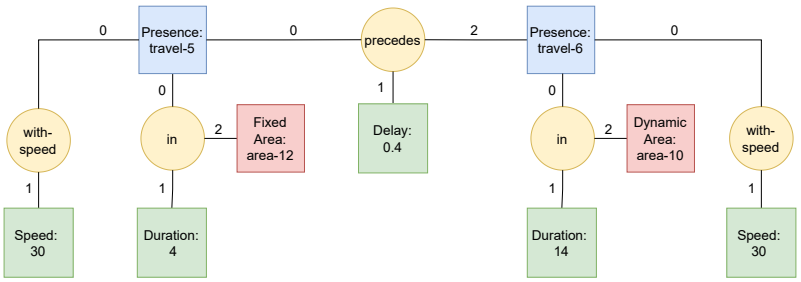


FIGURE 6.2 – Exemple de graphes conceptuels de trajectoire

en section 6.2.1, page 145 comment nous les traitons pour ce cas d'application.

La figure 6.2 donne un exemple de trajectoire sous forme de graphe conceptuel, où deux détections successives ont été effectuées, constituant deux unités d'information connectées. Elle représente la détection d'un véhicule dans l'*area-12* de type *FixedArea* durant un temps de valeur *4*, à une vitesse de valeur *30*, puis, après un temps de valeur *0.4*, le même véhicule est détecté dans l'*area-10* de type *DynamicArea* durant un temps de valeur *14* à une vitesse de valeur *30*.

6.1.2 Scénarios

3 scénarios ont été choisis pour être présentés ici, car ils sont représentatifs des autres scénarios. Les résultats obtenus par les différents algorithmes sur ces 3 scénarios sont étudiés dans la section 6.2.3, page 149.

Le premier scénario considéré, représenté figure 6.3, a été présenté comme exemple au début de la section 6.1.1, page 140 qui précède. Il correspond à la surveillance d'une région côtière, et comprend 2 graphes constitués chacun de 6 nœuds et 5 arcs.

Ce scénario est certes simple mais sert de base de comparaison pour les autres scénarios, et a de plus permis d'obtenir les résultats de la section 6.2.5, page 155, comme motifs communs à tous les scénarios.

Le second scénario, illustré sur la figure 6.4, est un cas plus complexe où 7 véhicules contournent une île pour atteindre leur objectif. Leurs points de départ sont les lignes jaunes et leur point d'arrivée la ligne rouge. Les lignes vertes représentent les 7 trajectoires de véhicules. Par ailleurs 2 des zones sont surveillées par des radars fixes et 3 par des drones. Ce scénario est modélisé en 7 graphes de 10 nœuds et 9.6 arcs en moyenne.

Le troisième scénario, représenté figure 6.5, est un cas de surveillance côtière où 25 trajectoires de véhicules sont détectées, dont les points de départ sont les côtes surlignées par une ligne orange et les points d'arrivée sont le long de la ligne orange dans l'eau. 6 zones sont surveillées par drone mobile et 2 par des radars fixes. La base correspondante contient 25 graphes de 6.8 nœuds et 6.0 arcs en moyenne.

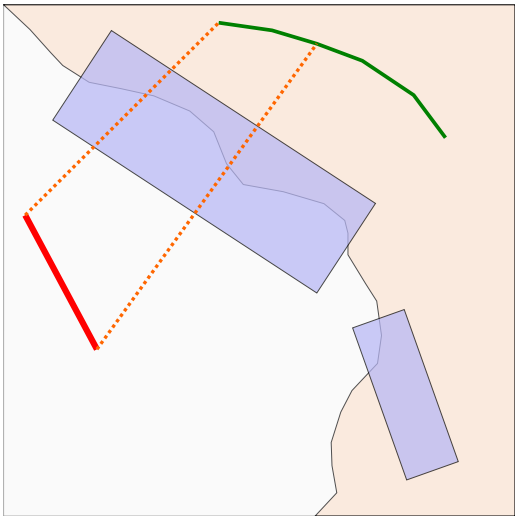


FIGURE 6.3 – Représentation du scénario 1 où 2 zones sont surveillées par radar fixe et où 2 trajectoires de véhicules sont détectées.

6.1.3 Critères

Dans les expériences menées en section 6.2, page 144, 4 critères numériques et deux critères qualitatifs sont étudiés.

Le premier critère numérique est le nombre de motifs renvoyés, noté *numberOfPatterns*. Il faut qu'il soit non nul afin d'obtenir des motifs caractéristiques du scénario, et il est attendu que *cgSpan-n*, *cgSpan-s* et *cgSpan-ns* réduisent la quantité de motifs tout en garantissant une meilleure qualité de résultat par rapport à *DMGM-GSM*.

Le second critère numérique est le temps d'exécution, noté *miningDuration*. Notre objectif est de minimiser ce critère, afin de prouver la possibilité d'utiliser *cgSpan* en temps réel dans ce contexte.

Les troisième et quatrième critères numériques sont la taille moyenne des motifs *meanPatternSize*, en nombre de nœuds, et leur support moyen *meanSupport*. Notre objectif avec ces critères est de caractériser les motifs identifiés par les différents algorithmes pour permettre leur comparaison sur la réponse apportée à la problématique métier.

Enfin, deux critères qualitatifs sont discutés lors de l'analyse des différents résultats : le premier critère est l'adéquation entre scénario considéré et les motifs obtenus d'une part ; le second critère est l'évaluation de la spécificité des motifs à un scénario, c'est-à-dire à quel point ces motifs se différencient des motifs des autres scénarios.

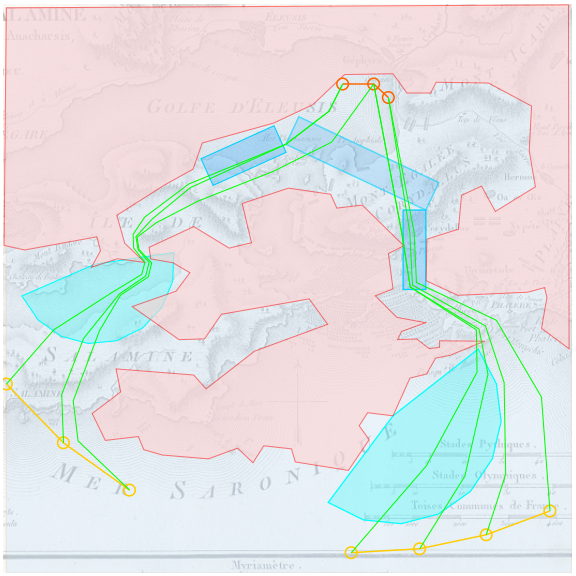


FIGURE 6.4 – Représentation du scénario 2 où 2 zones sont surveillées par radar fixe et 3 par des drones mobiles.

6.2 Résultats

Cette section présente les résultats des expériences menées. Les paramètres utilisés sont le seuil de support, *minSupport*, qui varie de 0.4 à 1, et la taille maximale de motif en nombre d'arcs, *maxEdgeCount*, qui varie de 1 à 7. Pour ces expériences, une valeur de *maxEdgeCount* supérieure à 7 ou une valeur de *minSupport* inférieure à 0.4 conduit à un échec mémoire sur certains scénarios. Ce point est discuté dans la section 6.2.3, page 149.

La section 6.2.1 introduit brièvement l'enrichissement du vocabulaire effectuée ainsi que les choix concernant la traduction des graphes conceptuels en graphes étiquetés pour cette application. Les résultats sont présentés dans les trois sections qui suivent : la section 6.2.2, page 147 compare les résultats obtenus par rapport à *DMGM-GSM* sur les 9 scénarios, la section 6.2.3, page 149 décrit une étude de l'influence des paramètres sur les deux premiers critères, et la section 6.2.4, page 151 détaille des exemples de motifs renvoyés. Enfin, la section 6.2.5, page 155 propose un calcul pour déterminer la complexité en mémoire, motivé par le problème rencontré sur les plages de valeurs de paramètres évoqués ci-dessus.

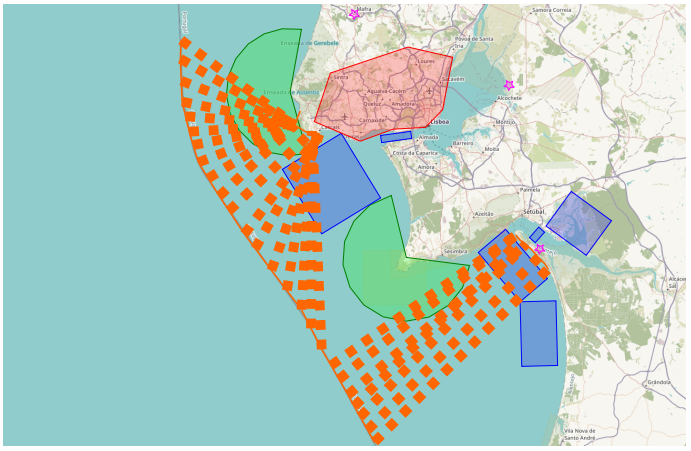


FIGURE 6.5 – Représentation du scénario 3 où 6 zones sont surveillées par drone mobile et 2 par des radars fixes.

6.2.1 Enrichissement et interprétation des connaissances

Ajout de signatures dans le vocabulaire

Afin de permettre de valoriser un point crucial de *cgSpan*, à savoir la prise en compte des signatures et des règles d'inférence pour maximiser les critères de non-redondance des motifs et d'efficacité, présentés chapitre 5, section 5.3.2, page 135, il nous faut disposer de signatures et de règles d'inférence. Sinon, on ne peut utiliser que *cgSpan-n*, ce qui limite l'intérêt de la solution proposée.

Nous proposons d'utiliser *cgSpan-n* sur les données fournies pour identifier des motifs fréquents de brique élémentaire, et ainsi déterminer les signatures effectivement respectées par la majorité des connaissances. Dans un second temps, ces signatures potentielles sont comparées à toutes les occurrences des relations dans les graphes conceptuels pour vérifier leur validité. Enfin, la signature la plus restrictive, dont les contraintes sont respectées par tous les graphes conceptuels, est gardée.

Par exemple, nous avons trouvé comme motif fréquent la brique élémentaire suivante :

$$with - speed(Presence, Speed)$$

Ensuite, nous avons vérifié en comparant dans chaque graphe conceptuel, de chaque scénario, si pour chaque occurrence de *with-speed*, le type des concepts connexes est respectivement égal ou plus spécifique que *Presence* et *Speed*, ce qui n'est pas le cas pour cet exemple.

Enfin, nous avons comparé les différentes signatures potentielles de *with-speed* ainsi obtenues, à savoir la précédente ainsi que : *with-speed(Presence, Value)*, *with-speed(Entity,*

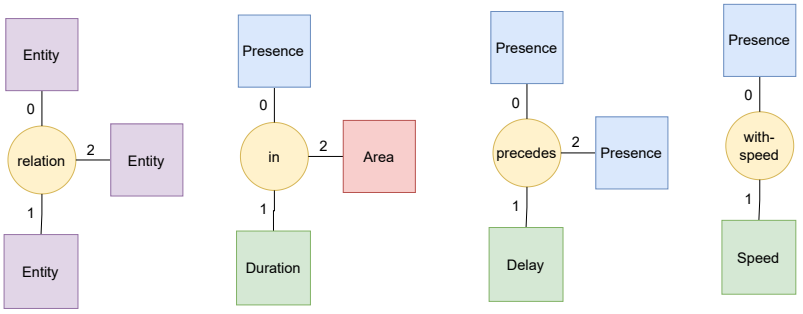


FIGURE 6.6 – Signatures des relations déduites d'une analyse des jeux de données

Speed), *with-speed(Presence, Entity)* et *with-speed(Entity, Entity)*. *with-speed(Presence, Speed)* étant la plus restrictive, c'est celle qui a été retenue.

Les signatures obtenues des relations *with-speed*, *in*, *precedes*, et *relation* d'arité 3 (celle d'arité 2 ayant comme signature *relation(Entity, Entity)*) sont représentées figure 6.6.

Traitements supplémentaires

Deux prétraitements concernant respectivement les valeurs numériques et les marqueurs individuels sont effectués.

Valeurs numériques Pour la traduction des graphes conceptuels dans le formalisme de DMGM-GSM, les traitements sont identiques à ceux décrits dans la section 5.2, page 127, à une différence près : les valeurs présentes dans les données sont fusionnées avec les types renseignés et ajoutées dans leur chemin taxonomique comme spécialisation de ces types. Ainsi, le nœud [*Speed* : 30] a pour chemin taxonomique *Entity_Value_Speed_Speed :30*

Ce traitement particulier vise à retrouver des valeurs dans les motifs tout en gardant la sémantique du type. Par conséquent, il est assuré que, par exemple, une occurrence de vitesse ne peut supporter une occurrence de durée de même valeur absolue : leur type doit également être identique.

Marqueurs individuels Pour *cgSpan-n* et *cgSpan-ns*, lors de la transcription depuis un motif de briques élémentaires vers un motif de graphe conceptuel, une information supplémentaire est gardée sur les étiquettes des arcs connectant deux briques élémentaires. Cette information ajoutée est nécessaire car les valeurs ne constituent pas un marqueur individuel.

En effet, comme indiqué dans la section 5.2.2, page 130, le chemin taxonomique des nœuds concepts communs à deux briques élémentaires constitue l'étiquette d'un arc les reliant. Ainsi, dans la figure 6.2, un arc dont l'étiquette est *Entity_Presence_travel-5* re-

	cgSpan-n	*cgSpan-s*	*cgSpan-ns*	*DMGM-GSM*
numberOfPatterns	0.53	0.08	0.0069	22 889.5
miningDuration	0.7	0.49	0.37	0.64
meanSupport	0.9	1.02	0.99	0.94
meanPatternSize	2.02	0.93	1.92	4.35

TABLE 6.1 – Résultats relatifs à *DMGM-GSM* sur les critères de *numberOfPatterns*, *miningDuration*, *meanSupp* et *meanPatternSize* moyennés sur les 9 scénarios et chacune des 49 combinaisons de valeurs de paramètres définies. La dernière colonne donne les valeurs absolues.

lierait les deux briques élémentaires obtenues à partir des relations *with-speed* et *in* le plus à gauche. Cependant, les valeurs étant traitées comme des marqueurs individuels lors de la construction d'un motif, si les deux nœuds concept d'étiquette *Speed :30* sont dans le même motif, leurs briques élémentaires respectives sont interconnectées par une étiquette *Entity_Value_Speed_Speed :30*. Par conséquent, lors de la reconstruction en motif de graphe conceptuel, ces deux nœuds seront fusionnés. Nous avons alors modifié l'information stockée dans les arcs de graphes de briques élémentaires pour éviter cela, et reconstruire le bon motif. D'une part, au lieu de simplement comprendre le chemin taxonomique d'un nœud en commun, l'étiquette d'un arc comprend également le rang du nœud dans les arguments des relations connexes. D'autre part, si deux nœuds de même valeur ou marqueur sont distincts dans les graphes conceptuels transformés en briques élémentaires, un identifiant unique est ajouté afin de les différencier.

6.2.2 Comparaison entre DMGM-GSM et cgSpan

Cette section présente des résultats généraux moyennés sur les 9 scénarios, dont les 3 présentés en section 6.1.2, page 142. Les valeurs des 4 critères, définis en section 6.1.3, page 143, sont calculés de deux manières : *numberOfPatterns* et *miningDuration* sont calculés pour chaque scénario, chaque combinaison des valeurs de paramètres telles que définies en début de section 6.2, page 144 (49 combinaisons au total), répété 3 fois ; *meanSupp* et *meanPatternSize* sont calculés de la même façon, mais moyennés en plus sur l'ensemble des motifs de chaque exécution.

Le tableau 6.1 donne ces résultats par rapport aux résultats de *DMGM-GSM*. La dernière colonne donne les résultats numériques absolus obtenus par *DMGM-GSM* avec *miningDuration* en secondes, *meanSupp* en proportion d'occurrences par rapport au nombre de graphes conceptuels du jeu de données, et *meanPatternSize* en nombre de nœuds.

Un nombre étonnamment élevé de motifs et des disparités

Trois remarques sont formulées sur les caractéristiques générales des résultats d'extraction de motifs.

Pour le scénario 1 constitué de 2 graphes, selon les paramètres, *DMGM-GSM* identifie entre 1000 et 3000 motifs, *cgSpan-n* entre 4 et 36, *cgSpan-s* entre 200 et 850 et *cgSpan-ns* entre 1 et 15.

Alors que les bases associées aux trois scénarios sont respectivement constituées de 2, 7 et 25 graphes conceptuels entre 5 et 10 nœuds chacun en moyenne, et que les autres scénarios non présentés sont du même ordre de grandeur, certains scénarios donnent 200 motifs de taille 1 et jusqu'à 1 million de motifs de taille 8 ou 9.

La différence d'ordre de grandeur entre la taille des bases et le nombre de motifs renvoyés peut sembler surprenante : *DMGM-GSM* obtient 22889.5 motifs en moyenne sur une exécution ; les exécutions de *cgSpan-n* et *cgSpan-s* suivent un ordre de grandeur de 2000 à 10000 motifs construits, sachant que le scénario le plus grand en terme de nombre de graphes et leur taille respective est le 2, avec 25 graphes de taille 7 en moyenne.

Ensuite, il existe une forte disparité entre les différents algorithmes sur le nombre de motifs, allant jusqu'à un facteur 10000 en moyenne entre *cgSpan-ns* et *DMGM-GSM*.

Enfin, en regardant les résultats au cas par cas, alors que la durée moyenne d'une extraction de motifs dans un tel cas appliqué est de l'ordre de 20 à 30 ms, certains cas dépassent les 20 secondes voire mènent à un échec mémoire. Comme évoqué précédemment, l'échec mémoire survient avec certains scénarios lorsque *maxEdgeCount* dépasse 7. Il existe ainsi des cas limites à l'utilisation de ces algorithmes.

Explications Ces trois remarques s'expliquent par le fonctionnement de *DMGM-GSM* qui explore tous les motifs possibles et dont l'implémentation fournie ne dispose pas du calcul de signification statistique, permettant de discriminer une partie des motifs. *cgSpan* remédie au problème par son critère de non-redondance, et par sa politique d'étiquetage des briques élémentaires combinées aux signatures qui permet de diminuer l'espace de recherche. Toutefois, nous sous-estimons malgré tout l'espace des motifs possibles, y compris dans le scénario 1 pourtant très simple. Un calcul de cette complexité est proposé en section 6.2.5, page 155 pour conclure sur ces questionnements.

Des motifs caractérisant les algorithmes

Deux remarques sont à formuler sur les caractéristiques des motifs en particulier.

D'abord il existe une relation claire entre la taille moyenne d'un motif et l'utilisation ou non du module de briques élémentaires : le tableau 6.1 montre que les algo-

rithmes *cgSpan-n* et *cgSpan-ns* construisent des motifs en moyenne deux fois plus grands que *cgSpan-s* et *DMGM-GSM*.

Une seconde remarque est que le support moyen ne semble pas varier d'une méthode à l'autre. Cela semble étonnant vu les différences sur les autres critères.

Explications La première remarque s'explique par les exemples de motifs étudiés en section 6.2.3 : les motifs sont deux fois plus grands quand le module de briques élémentaires est utilisé car dans le cas contraire, beaucoup de motifs correspondant à une brique élémentaire partielle sont construits. Ces motifs correspondent à une relation binaire et un seul de ses arguments, comme *with-speed(-,Speed)*, et une relation ternaire et un seul ou deux de ses arguments, tel que *precedes(-,Presence,-)*.

La seconde remarque s'explique par les caractéristiques des scénarios, qui comprennent des instances de *Presence*, c'est-à-dire la détection d'un véhicule à un endroit donné, et des instances de *MonitoredArea*, les zones surveillées, auxquelles sont systématiquement rattachées des caractéristiques sur la vitesse et le temps depuis la dernière détection. Il y a une homogénéité des données fournies, où très peu de relations sont présentes, seulement 3, et se retrouvent dans presque tous les graphes. Chaque graphe est ainsi constitué d'un ensemble de 1 à 3 nœuds concept *Presence* reliés par la relation *precedes*, tel que représenté dans la figure 6.2, page 142, avec leurs caractéristiques associées renseignées par les relations *in* et *with-speed*.

Ainsi, il n'y a que très peu de variation d'un graphe conceptuel à l'autre, et la majorité, comme l'utilisation de *cgSpan-s* et surtout *cgSpan-ns* le montrent, correspondent à des signatures de relation. Cette connaissance sur la manière de construire les traces de trajectoires a pu être déduite des motifs.

Ils constituent une connaissance ontologique sous forme de règles d'inférences comme : *Si le graphe comporte un nœud de type* Presence, *alors celui-ci est connecté à un nœud de type* MonitoredArea *et un nœud de type* Duration *par la relation* in.

Il serait intéressant d'utiliser les règles d'inférence obtenues suite à ces analyses pour renvoyer des résultats différents, plus intéressants, en particulier non-redondants avec ces connaissances acquises. Cela mènerait ces travaux applicatif depuis une tâche d'extraction de motifs vers une tâche d'extraction de règles d'associations.

6.2.3 Influence des paramètres

Notre objectif est d'observer l'influence des paramètres sur les deux premiers critères sur es 3 scénarios présentés section 6.1.2, page 142. *minSupport* et *maxEdgeCount* sur les critères de *numberOfPatterns* et *maxEdgeCount*.

Les critères de *numberOfPatterns* et *miningDuration* sont calculés pour des valeurs

de *minSupport* et de *maxEdgeCount* qui varient respectivement de 10% et 1 arc. Dans les figures 6.11, page 159, à 6.14, page 162, *numberOfPatterns* et *miningDuration* sont représentés en fonction de *minSupport* avec *maxEdgeCount* fixé à 7, puis en fonction de *maxEdgeCount* avec *minSupport* fixé à 0.8 pour les 3 scénarios de référence. Ces critères sont choisis car pour des valeurs supérieures de *maxEdgeCount* et des valeurs inférieures de *minSupport*, un des quatre algorithmes ne termine pas suite à une explosion mémoire. Le problème lié à cette complexité est étudié en section 6.2.5, page 155.

Les six figures 6.11, page 159 et 6.12, page 160 confirment les résultats relatifs de la section 6.2.2 : les motifs sont nombreux par rapport à la taille des jeux de données et les algorithmes renvoient des résultats d'ordre de grandeur différents.

Les graphes de la figure 6.11 montrent une transition importante de *numberOfPatterns* quand *minSupport* passe les seuils 0.7 et 0.8 dans les deux premiers scénarios respectivement, et dans une moindre mesure 0.6 dans le scénario 3, pour les algorithmes *DMGM-GSM* et *cgSpan-s*. Sur le reste des courbes, les variations sont peu importantes : ces deux observations tendent à montrer qu'il existe deux classes de motifs pour les trois scénarios, où chaque classe réunit des motifs de support équivalent.

Une troisième observation est qu'il reste un nombre réduit de motifs suite à ces transitions pour les algorithmes *cgSpan-n* et *cgSpan-ns*, égal à 4 et 1 respectivement.

Les graphes de la figure 6.12, page 160 apportent l'information que les rapports moyens entre les algorithmes du tableau 6.1, page 147, et des 2 premières remarques formulées en section 6.2.2, page 147, se vérifient lors de la variation des paramètres.

De plus, sans l'utilisation du module de briques élémentaires, on observe, comme attendu, par la progression de *numberOfPatterns*, que le coût en mémoire est surtout lié au paramètre *maxEdgeCount* : alors que *cgSpant-n* et *cgSpan-ns* ont un *numberOfPatterns* stable, *DMGM-GSM* et *cgSpan-s* ne font qu'augmenter. On se limite ici à un ordre de grandeur de 1000 motifs maximum, quel que soit le scénario, cela semble se stabiliser pour des valeurs plus importantes de *maxEdgeCount*.

Les graphes de la figure 6.13, page 161 montrent d'une part que les transitions évoquées précédemment impactent également l'efficacité des algorithmes, surtout dans les scénarios 1 et 2. Cela est d'autant plus important dans le scénario 2, où *DMGM-GSM* passe d'un rapport 4 à un rapport 1.3 avec *cgSpan-s* en terme de *miningDuration*.

Ces graphes mettent d'autre part en avant le fait que *cgSpan* et ses variantes sont bien plus stables que *DMGM-GSM* sur le plan de l'efficacité.

Enfin, les graphes des figures 6.14, page 162 sont les plus instables. Une erreur s'est produite pour le traitement de cgSpan-n sur le scénario 2, correspondant à une valeur exceptionnellement élevée. La tendance générale qui ressort des trois scénarios est le rapide accroissement de *miningDuration* pour *DMGM-GSM*, et *cgSpan-s* dans une moindre

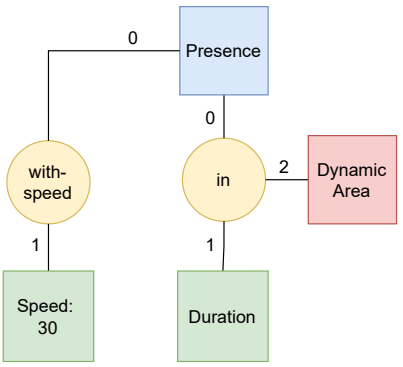


FIGURE 6.7 – Motif renvoyé par *cgSpan-ns* sur les trois scénarios

mesure.

La courbe du scénario 2 pour *cgSpan-n* semble être une erreur après comparaison avec le temps d'exécution pour ces paramètres sur plusieurs autres occurrences.

6.2.4 Motifs renvoyés

Dans la suite, nous appelons motifs *le plus informatif* ceux renvoyant le plus d'information selon deux caractéristiques : au niveau de la structure, ce sont les plus grands, et donc recouvrent le plus d'instances possibles ; au niveau des étiquettes, ce sont les plus spécialisés tout en restant fréquents au regard des paramètres choisis.

Ces motifs sont identifiés en classant les motifs renvoyés par les algorithmes selon leur taille puis selon leur degré de généralisation (calculé comme une distance, dans les hiérarchies, entre les étiquettes des motifs et la racine des hiérarchies).

Des motifs caractérisant les scénarios

Scénario 1 Le motif de *cgSpan-ns* restant suite aux transitions discutées précédemment correspond à celui représenté figure 6.7. Il correspond à une unité d'information de trajectoire, caractéristique de la modélisation sous-jacente des données.

En plus des connaissances inhérentes aux signatures, qui sont discutées dans la section 6.2.1, page 145, et aux règles d'inférence sous-jacentes à la façon dont les données sont construites, discutées en fin de section 6.2.2, ce motif indique que la plupart des détections ont été effectuées par des drones pour des véhicules d'une vitesse de 30.

Scénario 2 Pour le scénario 2, les motifs les plus informatifs construits correspondant à 2 à 3 unités d'informations qui se succèdent, chacune équivalente au motif en figure 6.7.

Pour le scénario 3, ce sont des motifs de 1 à 2 unités d'informations qui se succèdent, également chacune équivalente au motif en figure 6.7. La seule différence entre ces motifs et celui du scénario 1, en plus du nombre d'unités d'informations, est que le type de la zone renseignée dans les unités supplémentaires n'est pas le même. Ils sont similaires en cela au graphe conceptuel de la figure 6.2, page 142, où on aurait enlevé toutes les étiquettes de valeurs et marqueurs excepté pour les nœuds [*Speed* : 30].

Le scénario 2 a, dans son motif *le plus informatif*, le motif de la figure 6.7, répété 3 fois, avec la première unité renseignant une *FixedArea*, et les deux suivantes une *DynamicArea*. C'est exactement ce qui est attendu après observation de la figure 6.4, page 144 qui représente le scénario, où l'on peut remarquer que 4 des 7 trajectoires passent par 1 zone surveillée par un radar fixe puis 2 zones surveillées par des drones. Ce motif, et ses variantes qui le généralise, sont retrouvés pour une valeur de *maxEdgeCount* supérieure ou égal à 15 et une valeur de *minSupport* inférieure ou égale à 0.5 (et sûrement jusque 0.57, soit environ 4/7, le nombre de trajectoires qui correspondent), parmi 21 576 motifs. *cgSpan-s* les retrouve également, mais pour une valeur de *maxEdgeCount* supérieure ou égal à 20 et une valeur de *minSupport* inférieure ou égale à 0.5 parmi 696 387 motifs, et parfois cela résulte en un échec mémoire.

Les différences dans les valeurs de paramètres nécessaires sont dues à la formalisation sous forme de brique élémentaires du premier module de *cgSpan*, qui réorganise les graphes conceptuels. Les différences seraient bien plus importantes si la taille des motifs maximale était calculée en fonction du nombre de nœuds, et non pas en fonction du nombre d'arcs comme dans le cas présent avec *maxEdgeCount*. Les détails de cette remarques sont étudiées par un calcul effectué en fin de section 6.2.5, page 155.

Scénario 3 Le motif *le plus informatif* du scénario 3 est similaire à ceux du scénario 2 : il correspond à deux unités d'information de trajectoire ; celle dont la zone est de type *DynamicArea* précède l'unité dont la zone est de type *FixedArea*. Il est retrouvé avec des valeurs de *minSupport* inférieure ou égale à 0.1, et des valeurs de *maxEdgeCount* supérieures ou égales à 7 pour les algorithmes *cgSpan-n* et *cgSpan-ns*, et des valeurs de *maxEdgeCount* supérieures ou égales à 13 pour les deux autres algorithmes.

Discussion des résultats Ainsi, les scénarios ont pour motif commun celui en figure 6.7, et se différencient par le nombre et les types de zones traversées, ainsi que par les caractéristiques de ces motifs, de support 0.57 et 0.12 pour respectivement le motif du scénario 2 et le motif du scénario 3.

Le tableau 6.2 caractérise chacun des scénarios par la description de leur motif *le plus informatif* discuté ci-dessus. La taille est donnée en nombre de nœuds, noté NdN. La dernière colonne renseigne sur les types de zones traversées, sous forme d'une séquence qui

Scénario	**Taille (NdN.)**	**Support (%)**	**Types zones**
1	6	1	d
2	13	0.57	f-d-d
3	20	0.12	d-f

Table 6.2 – Caractéristiques de chacun des motifs *le plus informatif* caractérisant les scénarios, et discuté en section 6.2.3

indique le type des zones traversée et l'ordre de leur traversée : par exemple, *f-d* indique qu'une *FixedArea* a été traversée puis une *DynamicArea*. Dans les trois cas, les motifs caractérisent les scénarios dont ils sont extraits selon les différentes caractéristiques représentées : leur taille, leur support, le type des zones traversées et l'ordre dans lequel elles le sont. De plus, ils sont en adéquation avec la représentation des scénarios déjà introduite.

Ces deux critères qualitatifs peuvent surtout être confirmés pour les scénarios 2 et 3, le premier étant bien trop simple pour permettre des conclusions satisfaisantes.

Un élément qui permettrait de compléter ces caractéristiques serait de compléter ces séquences, en utilisant les valeurs de *Delay* qui séparent chaque détection d'une trajectoire de véhicule donnée, et en utilisant les valeurs de *Duration*. Actuellement elles diffèrent toutes, et il faudrait donc trouver un moyen de les comptabiliser et abstraire leur information. Une piste pour aller vers des motifs plus complets prenant en compte ces valeurs est discutée dans le bilan de ce chapitre.

Nous avons ainsi réussi à caractériser ces différents scénarios par un motif. La section qui suit étudie quelques autres motifs, notamment pour comparer les résultats des différents algorithmes, ainsi que pour observer les variantes de motifs.

Autres motifs

Les motifs construits par *cgSpan-n* incluent le motif de la figure 6.7, ainsi que ceux représentés dans le treillis de la figure 6.8. Ils ne semblent pas apporter d'information supplémentaire, si ce n'est que des véhicules de vitesse 30 sont souvent détectés dans des zones non-nécessairement surveillées par des drones, et réciproquement, des véhicules sont détectés par des drones avec une vitesse non-nécessairement égale à 30. Pour avoir plus de détails, il faudrait garder l'information des vitesses qui supportent le nœud de type *Speed*, pour se rendre compte de leur répartition.

L'étude des motifs identifiés par *cgSpan-s* et *DMGM-GSM* conduit à les classer en deux catégories. La première comprend des motifs non informatifs, soit parce qu'ils correspondent à des parties de voisinages de relation, et sont en cela incomplets, soit parce

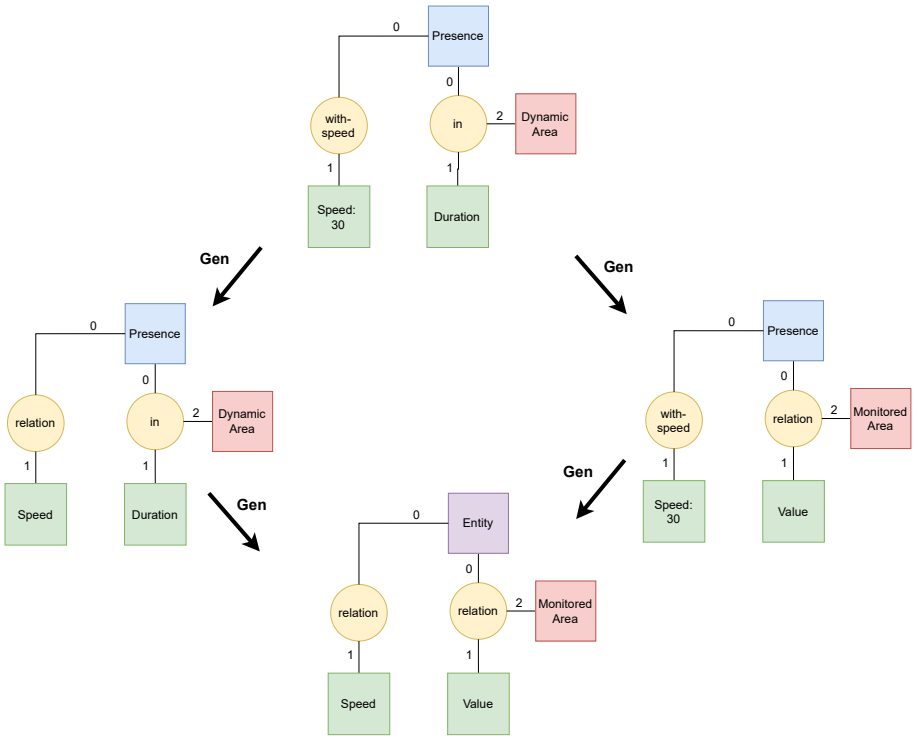


Figure 6.8 – Motifs identifiés par *cgSpan-n*

qu'ils sont trop généraux. La seconde catégorie contient des motifs d'intérêt supérieur, qui enrichissent les précédent en fournissant une valeur précise au sein d'un motif fréquent, offrant une information de la forme "Une vitesse de 30 est une vitesse fréquente".

La figure 6.9 renseigne des exemples de motifs non informatifs en tant que briques élémentaires partielles. Le fait que de certains de ces motifs ne se retrouvent pas comme sous-partie de *cgSpan-n* et *cgSpan-ns* s'explique par deux raisons. D'une part il est possible que le reste des briques élémentaires dont ils font partie ne soit pas fréquent. D'autre part, le fonctionnement du module de brique élémentaire avec *DMGM-GSM* peut les ignorer : par la politique actuellement appliquée d'étiquetage de brique élémentaire, toutes les sous-étiquettes de la briques sont spécialisées en même temps.

Ainsi, dans le chemin taxonomique d'une brique élémentaire, on passe du voisinage de relation *with-speed(Presence, Speed)*, au voisinage *with-speed(travel-0-0, Speed :30)* (où *travel-0-0* est un marqueur de type *Presence*). Or, les autres occurrences de cette valeur de 30 cooccurrent avec des marqueurs de type *Presence* différents de *travel-0-0*. Ainsi, contrairement à *DMGM-GSM* et *cgSpan-s* qui explorent toutes les combinaisons possibles de spécialisation, la spécialisation conjointe actuellement effectuée dans *cgSpan* fait perdre des informations dans ce cas.

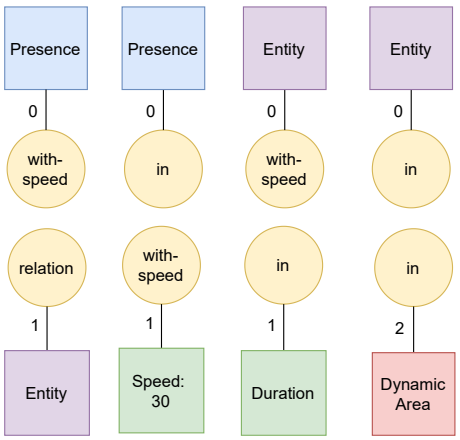


FIGURE 6.9 – Motifs renvoyés par *cgSpan-s* et *DMGM-GSM*

C'est un problème déjà évoqué dans l'article FACI et al., 2021b de même que dans la section 5.3.3, page 136 de ce manuscrit. En contrepartie, *cgSpan-n* et *cgSpan-ns* renvoient bien moins de motifs, mais il serait intéressant de formuler un politique d'étiquetage de brique élémentaire moins radicale.

6.2.5 Calcul de complexité

Cette section établit les bases d'un calcul de complexité afin de justifier les résultats obtenus dans les sections précédentes qui peuvent a priori sembler surprenants.

L'explosion de *numberOfPatterns* pour une base de 2 graphes du scénario 1 (ainsi que les deux autres) s'explique par le fait que le nombre de motifs possibles, pour un motif connu de taille fixée comportant k nœuds, avec *DMGM-GSM* est égal à :

$$\prod_{i}^{k} S_i$$

où S_i est le nombre de spécialisations du nœud i. Il faut ensuite sommer ces estimations pour des motifs de taille k variable.

Dans le scénario 1 par exemple, pour des graphes constitués de 6 nœuds et une profondeur de hiérarchie en moyenne de 4, sachant que les marqueurs sont comptés comme les feuilles de la hiérarchie, on obtient $4^6 = 4\,096$ motifs potentiels environ. L'ordre de grandeur est ainsi cohérent avec les 1000 voire 3000 trouvés.

Dans le scénario 2, on obtient $4^{10} = 1\,048\,576$ pour un maximum de 176 000 motifs environ avec *DMGM-GSM*, et dans le scénario 3 on obtient $4^7 = 13\,125$ contre au maximum environ 1500 motifs avec *DMGM-GSM*. Les scénarios 2 et 3 sont moins proches de l'estimation, et cela peut s'expliquer par le fait que leurs graphes ont en moyenne une

densité (Wörlein et al., 2005 ; Cakmak & Ozsoyoglu, 2008) égale à 0.25 et 0.19 respectivement, contre 0.28 pour le scénario 1 : on observe une corrélation entre densité et proximité avec l'estimation. Par ailleurs, cela peut également s'expliquer par le manque d'homogénéité des graphes de la base, que partagent que très peu de motifs fréquents.

Concernant maintenant le calcul de taille d'un motif donné avant et après traduction en graphe de briques élémentaires, la traduction actuelle divise le nombre de nœuds d'un ordre de grandeur dépendant de l'arité des relations. Prenons le cas simple où le graphe conceptuel traduit n'a que des relations d'arité $n-1$, et qu'il n'est composé que de relations et leur voisinages isolés, c'est-à-dire qu'aucune relation n'a de nœud concept en commun. Alors la taille du graphe de briques élémentaires conceptuelles correspondant sera n fois moins importante, en nombre de nœuds, et aura une taille de 0 en nombre d'arcs.

Par ailleurs, plus les voisinages de relation ont des nœuds en commun, moins la taille, en nombres de nœuds ou d'arcs est réduite. Par exemple, soit un graphe conceptuel composé de k relations d'arité $n-1$, et où il y a m occurrences d'un concept communs à deux relations. Cela signifie qu'un nœud concept commun à 3 nœuds relations compte 3 fois, et qu'un nœud concept commun à 4 relations compte 6 fois, et dans le cas général, qu'un nœud concept commun à v relations compte $v(v-1)/2$.

La taille d'un graphe conceptuel est alors, en nombre de nœuds, égale à $n*k-m$ avant la traduction, et égale à k après traduction. En nombre d'arcs, sa taille passe de $k*(n-1)$ à m. Nous avons ainsi une manière de savoir quelle sorte de taille de graphe évolue en fonction de sa connexité. Cela pourrait être utilisé dans *cgSpan* pour développer un module de brique élémentaire prenant en compte ces calculs pour adopter une politique différente d'étiquetage de brique élémentaire. Ces résultats peuvent également être utilisés dans ce cadre applicatif pour permettre de comprendre les différences entre les algorithmes concernant les motifs renvoyés.

Ces résultats permettent de valoriser *cgSpan* sur une efficacité mémoire en plus de l'efficacité en temps déjà étudiée. Il faudrait par ailleurs prendre en compte la connexité des motifs dans l'estimation.

6.3 Bilan

En conclusion, les résultats montrent le caractère fonctionnel des modules de *cgSpan* dans un cas industriel appliqué à la caractérisation de trajectoires, réaffirment l'équilibre à trouver entre précision, concision et qualité des résultats, ainsi qu'efficacité de l'algorithme. Leur objectif de réduire le nombre de motifs à construire pour gagner en efficacité a été prouvé sur des données synthétiques dans le chapitre 5, et ici cette validation est

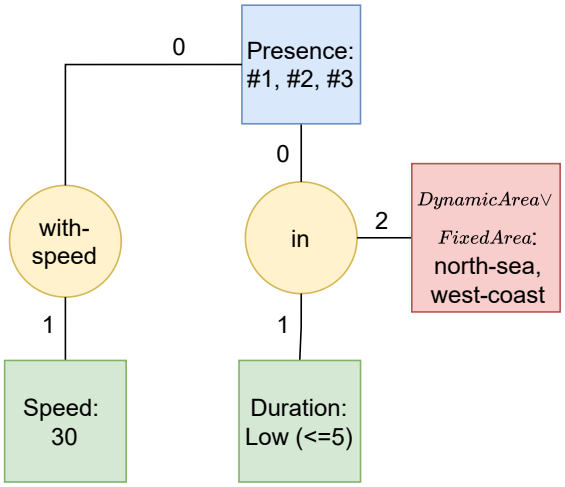


FIGURE 6.10 – Résultats préliminaires de motifs flous sur le cas d'application

confirmée sur des données réelles.

En outre, *cgSpan* a bien répondu au besoin métier exprimé, en caractérisant les différents scénarios. Nous avons ainsi extrait comme motif de trajectoire commun l'unité d'information, tel que représentée figure 6.7. Le choix de classer les motifs en fonction de leur degré de généralisation et de leur taille a permis de trouver des motifs spécifiques à chaque scénario parmi des dizaines voire centaines de milliers d'autres. Ainsi, au delà de caractériser par des critères les motifs retrouvés pour les élaguer, comme cela est fait dans *cgSpan*, viser de plus à classer ces motifs est une perspective motivante au vu des résultats sur ce cas d'application.

D'autre part, les motifs renvoyés ont permis d'induire des signatures de relation, puis dans l'itération présentée ici, de déduire des règles d'inférence. Cela offre des perspectives en terme d'interface homme-machine, dans une application, où un utilisateur classera les motifs selon qu'ils correspondent ou non à des signatures ou règles d'inférences, améliorant les résultats futurs. Par ailleurs, explorer le cas des algorithmes d'extraction de motifs contrastifs permettrait de directement sélectionner les motifs caractéristiques d'un scénario.

Enfin, des calculs en section 6.2.5 offrent des perspectives en terme de formalisation de mesures d'estimation de complexité, pour un algorithme d'extraction de motifs dans des graphes conceptuels tel que *cgSpan*.

Pour aller plus loin, une piste serait d'enrichir l'expressivité des motifs identifiés, notamment en combinant les propositions théoriques de la partie I, tel qu'illustré sur la figure 6.10, obtenue en modifiant cgSpan : en comptabilisant les valeurs dans les motifs en exploitant l'expressivité des graphes conceptuels flous, notamment avec des valeurs

linguistiques ; en prenant en compte les marqueurs pour spécifier ceux supportant le plus un nœud ; et enfin en considérant des types disjonctifs, donnant plus d'information que la simple généralisation.

Par exemple, dans la figure, le nœud $[Duration : Low(<= 5)]$ qui renseignent sur les valeurs couvertes par cette étiquette. C'est une étiquette non floue qui a remplacé les valeurs de durée inférieures à 5 dans les données, il serait alors intéressant d'affiner ce travail pour se placer dans le cadre flou. Le nœud $[Presence : \#1, \#2, \#3]$ est construit sans prendre en compte ses marqueurs, mais ces derniers sont ceux les plus souvent retrouvés dans les graphes conceptuels qui le supportent. Enfin, $DynamicArea \vee FixedArea$, en tant que généralisation de *DynamicArea* et *FixedArea*, est plus précis que *MonitoredArea*

Ces perspectives permettent d'améliorer la richesse des motifs renvoyés, et par là l'interprétabilité, tout en soulevant des défis algorithmiques.

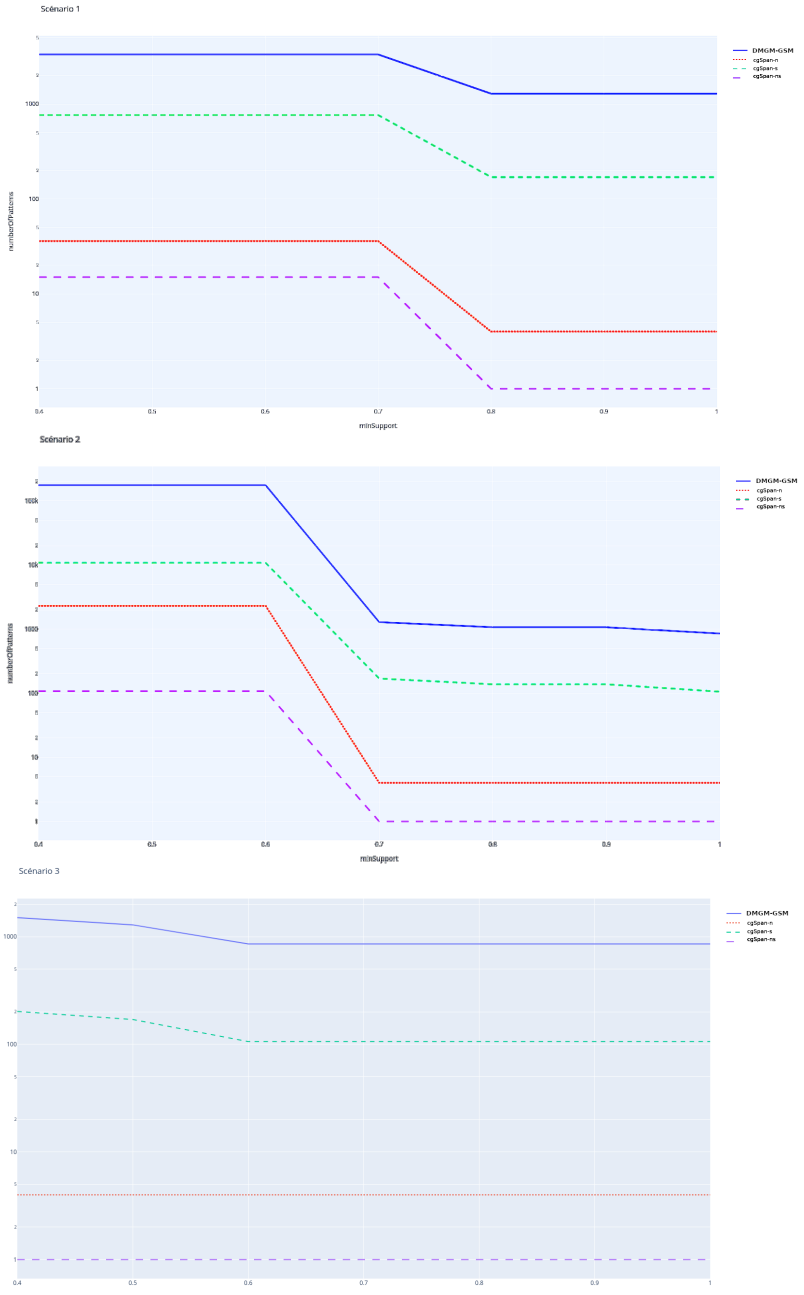


FIGURE 6.11 – Comparaison des 4 algorithmes selon *numberOfPatterns* (échelle log) en fonction de *minSupport*, avec une valeur de *maxEdgeCount* de 8, pour les trois scénarios de référence.
DMGM-GSM (en bleu), cgSpan-n (en rouge), cgSpan-s (en vert) et cgSpan-ns (en violet)

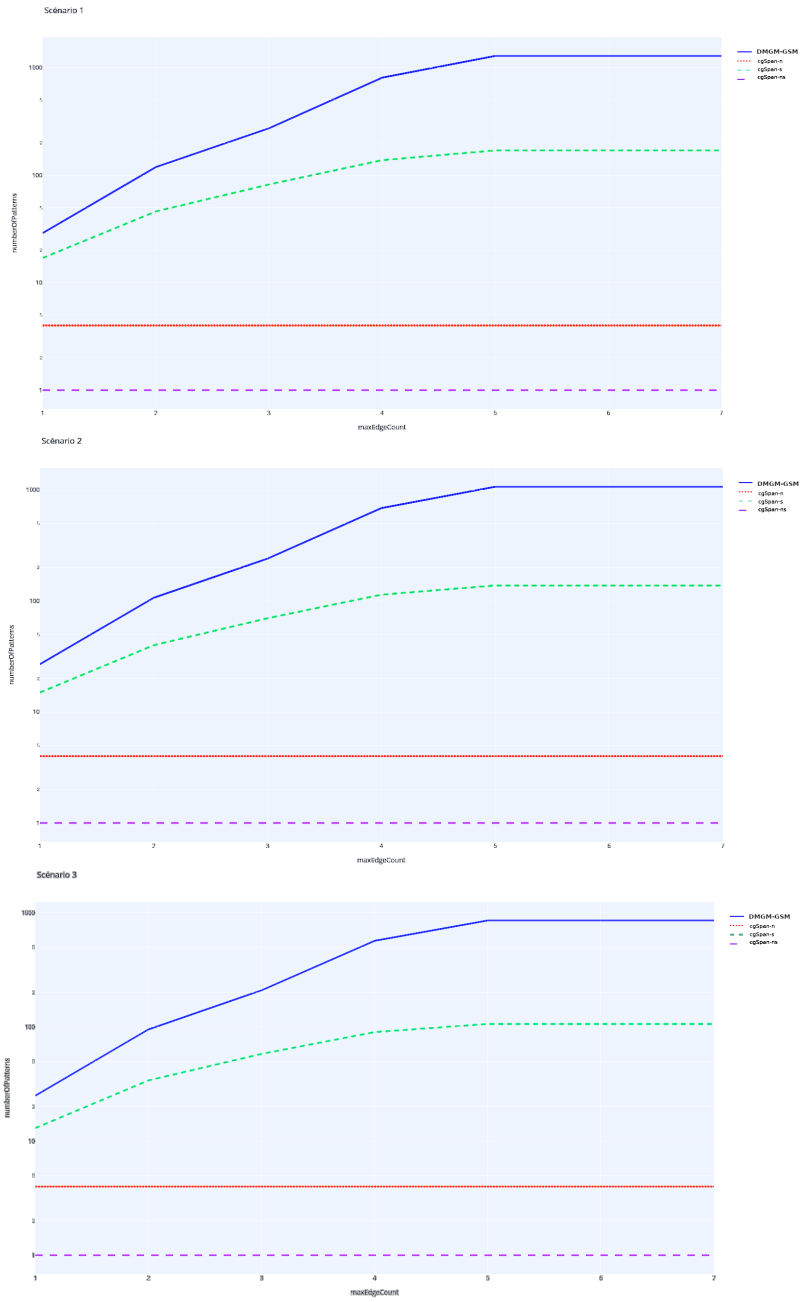


FIGURE 6.12 – Comparaison des 4 algorithmes selon *numberOfPatterns* (échelle log) en fonction de *maxEdgeCount*, avec une valeur de *minSupport* de 0.8, pour les trois scénarios de référence.
DMGM-GSM (en bleu), cgSpan-n (en rouge), cgSpan-s (en vert) et cgSpan-ns (en violet)

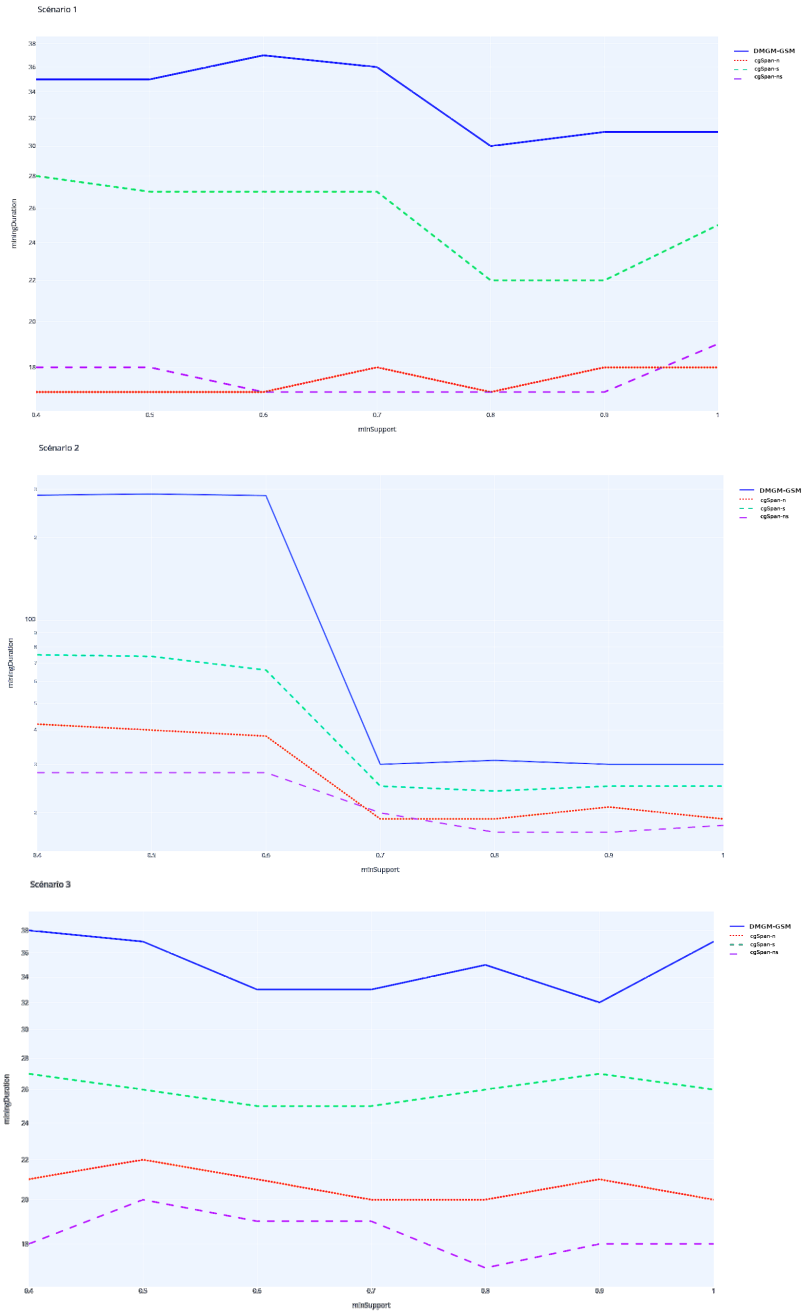


FIGURE 6.13 – Comparaison des 4 algorithmes selon *miningDuration* (en ms) en fonction de *minSupport*, avec une valeur de *maxEdgeCount* de 8, pour les trois scénarios de référence.
DMGM-GSM (en bleu), cgSpan-n (en rouge), cgSpan-s (en vert) et cgSpan-ns (en violet)

FIGURE 6.14 – Comparaison des 4 algorithmes selon *miningDuration* (en ms) en fonction de *maxEdgeCount*, avec une valeur de *minSupport* de 0.7, pour les trois scénarios de référence.
DMGM-GSM (en bleu), cgSpan-n (en rouge), cgSpan-s (en vert) et cgSpan-ns (en violet)

Conclusion et perspectives

Conclusion

Ce manuscrit aborde principalement trois questions : comment représenter, comment simuler et comment exploiter des connaissances dans le formalisme des graphes conceptuels.

Cette étude nous a conduit à effectuer une discussion argumentée sur les extensions floues des graphes conceptuels ; à proposer des algorithmes de simulation de bases de graphes conceptuels fondés sur différents modèles de connaissance ; à concevoir un algorithme efficace d'extraction de motifs fréquents non-redondants avec les connaissances ontologiques, et validé sur des données synthétiques du simulateur, ainsi que sur un cas d'application réel de caractérisation de trajectoires.

La représentation de connaissances sous forme de graphes conceptuels allie une grande expressivité issue de la logique des prédicats du premier ordre, une efficacité des raisonnements et des outils d'exploitation grâce aux algorithmes disponibles basés sur la théorie des graphes, et une interprétabilité importante de par leur forme graphique visuelle, intuitive dans les cas maîtrisés. La question de la représentation des connaissances sous forme de graphes conceptuels s'est posée, d'un point de vue théorique et pratique, afin d'évaluer les limites d'expressivité du formalisme, trouver des idées originales afin de dépasser ces limites et en évaluer les conséquences sur l'exploitation et l'interprétation. Nous avons proposé une étude comparative des graphes conceptuels flous (Faci et al., 2021c) sur les différents modèles de l'état de l'art, portant notamment sur la partie du formalisme impactée, les contraintes de représentation relâchées et les interprétations possibles. Il en résulte une vue d'ensemble de l'état actuel de ce qu'il est possible de représenter dans des graphes conceptuels flous, et de la portée de chaque contribution de l'état de l'art, avec ses avantages et ses inconvénients.

La simulation de connaissances sous forme de graphes conceptuels permet de disposer de bases de connaissances complexes, définies sur un modèle tel que des contraintes sous la forme de connaissances ontologiques ou implicites. Nous avons proposé *CG2A* (Faci et al., 2021a) comme algorithme de simulation de bases de graphes conceptuels dont les

critères de validation sont la variabilité et la prédictibilité : il fournit une réponse au vide de l'état de l'art, sous la forme d'un outil de génération de *benchmarks*, qui offrent la possibilité de comparer, sur des bases de référence communes, des algorithmes d'exploitation de connaissances sous forme de graphes conceptuels. En outre, la formulation des graphes conceptuels-γ de l'outil de simulation est conjointe à la proposition d'une brique élémentaire de l'algorithme d'exploitation : cela permet la validation de tels algorithmes d'exploitation par la mesure de la pertinence de leurs résultats sur des données synthétiques.

L'exploitation de connaissances sous forme de graphes conceptuels vise à raisonner sur les bases de graphes conceptuels afin d'en retirer des connaissances intéressantes au regard de certains critères. Nous l'étudions par la proposition d'algorithmes d'extraction de motifs d'intérêts ainsi que la définition de nouvelles formes de motifs. Nous avons proposé l'algorithme *cgSpan* (Faci et al., 2021b) comme premier algorithme d'extraction de motifs dans des graphes conceptuels, qui exploite trois de leurs spécificités. Il maximise la fréquence des motifs et leur informativité, comptée par une mesure de non-redondance avec les connaissances ontologiques et implicites. Un protocole expérimental a été établi, qui inclut une validation sur des *benchmarks* synthétiques généré au moyen de *CG2A* ainsi qu'une validation sur des données réelles dans un cadre industriel répondant au à un besoin métier de caractérisation de trajectoires pour la sécurité maritime.

Perspectives

Outre les propositions court terme évoquées dans les bilans des chapitres de ce manuscrit, des perspectives de recherche peuvent être définies basées sur les composantes théoriques, algorithmiques et pratiques de ce manuscrit. Elles sont présentées suivant l'organisation de ce manuscrit, selon qu'elles portent sur les aspects de représentation, simulation ou extraction.

Représentation de connaissances

Le chapitre 2 offre une discussion sur les modèles de graphes conceptuels flous de l'état de l'art pour la représentation de connaissances imprécises dans les graphes conceptuels, avec une emphase sur les questions de représentation.

Une piste pour la poursuite de ces travaux est d'examiner le cadre plus général des graphes conceptuels pondérés, où les poids peuvent être associés à d'autres sémantiques que celles des sous-ensembles flous : par exemple selon une sémantique probabiliste ou possibiliste, permettant de représenter des connaissances incertaines. Il n'existe pas à notre connaissance de tels formalismes de graphes conceptuels pondérés, et une pers-

pective serait alors d'adapter les travaux du chapitre 2 à ces différentes sémantiques.

Une piste qui suit est alors de combiner les différents modèles formalisés afin d'aboutir à un modèle complet de graphes conceptuels pour représenter des connaissances imparfaites, en incluant différentes sources d'imperfection. En particulier, un équilibre est à rechercher entre la certitude des informations représentées et leur précision, conduisant à un modèle fin de représentation de la connaissances.

Ensuite, la question des raisonnements dans de tels modèles, y compris dans le cadre flou, se pose et doit être formalisée. Les raisonnements proposés par les modèles de graphes conceptuels flous n'ont pas été discutés dans ce manuscrit, et une première étape consiste à effectuer une discussion comparative, pour, dans un second temps, en offrir une synthèse les combinant aux discussions présentées dans le chapitre 2. Cette prise de recul offre deux avantages au moins : les diverses d'interprétations détaillés dans les discussions du chapitre 2 pourront être enrichies d'une comparaison sur les mécanismes de raisonnement induits ; et les raisonnements dans les formalismes proposés de graphes conceptuels pondérés non flous pourront alors être définis de manière contrastive à ceux du cadre flou, résultant en une étude plus riche et plus complète.

De plus, d'autres perspectives sur la représentation dans un cadre flou sont encore à observer. Par exemple, en se basant sur la correspondance entre graphes conceptuels et hypergraphes conceptuels (Baget, 2003b), on peut définir des relations imparfaites (au sens logique) dont le voisinage et l'arité ne sont pas fixes. Dans le cadre des graphes conceptuels, cela pourrait se représenter par des relations dont les arcs qui les connectent à des concepts sont pondérés par l'importance du concept dans la relation. Dans le cadre des hypergraphes conceptuels, cela correspondrait à un hyperarc (qui est la traduction dans ce cadre des relations des graphes conceptuels) dont les concepts sont membres à un degré non binaire, conduisant ainsi à la définition d'une fonction d'appartenance pour chaque hyperarc sur l'ensemble des nœuds concept.

Une telle pondération pourrait par exemple être mise en œuvre dans des graphes conceptuels représentant des systèmes informatiques, par exemple pour représenter un degré d'utilisation d'une ressource : on peut envisager un nœud relation *utilisationRessource* qui pondère le lien entre la ressource (un fichier par exemple) et les processus qui l'utilisent. Cette pondération peut imposer une somme maximale correspondant à la capacité de la ressource a être utilisée en parallèle.

Une autre piste est de définir des règles d'inférence pondérées, en explorant la question de la position de la pondération. En effet, celle-ci peut porter sur les graphes conceptuels en prémisse et en conclusion, ou sur le mécanisme de déduction de la règle elle-même. D'autres part, une différenciation entre règle d'inférence, contrainte et règles de transformation pourra être effectuée pour distinguer leur sémantique spécifique.

Simulation de connaissances

Les travaux des chapitres 3 et 4, qui exploitent respectivement des ambiguïtés de traduction et des contraintes ontologiques, ont permis de proposer des outils de simulation de bases de graphes conceptuels. Ces propositions peuvent être enrichies de plusieurs manières.

Une première piste est d'envisager l'exploitation d'une autre forme de connaissance en entrée de la simulation : en particulier, comme suggéré dans la partie expérimentale du chapitre 5, on pourrait proposer une version de *CG2A* considérant des contraintes sous forme de fonctions de distribution. Ces dernières modéliseraient des contraintes sur les caractéristiques souhaitées des graphes conceptuels générés telles que leur taille, leur densité, les étiquettes utilisées ou les graphes conceptuels-γ qui les constituent par exemple.

Cette approche offre des avantages en termes de prédictibilité et de variabilité des bases générées : d'une part la prédictibilité serait enrichie des connaissances issues des fonctions de distribution utilisées, permettant de formaliser des résultats attendus plus finement ; d'autre part la variabilité, au lieu d'être définie selon une loi uniforme sur l'espace contraint par les connaissances ontologiques, suivrait alors les lois correspondants à ces fonctions, permettant ainsi un meilleur contrôle de ce paramètre.

Techniquement, pour inclure de telles fonctions dans *CG2A*, il faudrait remplacer les tirages aléatoires effectués selon une loi uniforme, à savoir le choix des graphes conceptuels-γ à combiner et l'affectation de leurs variables, par des choix effectués pour maximiser le respect de la fonction de distribution d'entrée par la base générée (par exemple, en minimisant une distance entre la distribution courante de la base et la fonction visée). Une autre stratégie consisterait à adopter une mise en conformité de la base par rapport à la fonction de distribution, par exemple à chaque ajout de graphe : cette mise en conformité serait par exemple une modification du graphe ajouté ou de l'entièreté de la base.

Une seconde piste est de proposer des outils de génération pour des graphes conceptuels plus complexes : en particulier on pourrait construire des bases de graphes conceptuels pondérés en profitant des contributions du chapitre 2. La mise en place de cette extension de *CG2A* repose sur la généralisation du concept de graphes conceptuels-γ proposé, que *CG2A* prend en paramètre d'entrée. Cette généralisation peut être effectuée par l'ajout d'une pondération telle que, par exemple, une valeur numérique entre 0 et 1 associée à certaines étiquettes, ou bien des variables linguistiques.

Ainsi, avec une distribution uniforme par exemple, une valeur serait associée à certaines étiquettes lors de l'étape d'affectation des graphes conceptuels-γ. Cela pose immédiatement des questions telles que comment assurer la cohérence des graphes suite à leur affectation (faut-il que chacune des valeurs possibles pour chaque étiquette soit compa-

tible avec celle des autres ? Ou est-il préférable d'ajouter une étape de mise en conformité suite à l'étape d'affectation ou de fusion ?) ou encore, comment combiner puis fusionner les graphe conceptuels γ pondérés entre eux.

Cette extension de *CG2A* pourrait également être mise en place par la modification de l'opérateur de fusion de graphes conceptuels proposée dans *CG2A* pour que chaque combinaison résulte en un graphe conceptuel pondéré. Par exemple, si deux nœuds concept de type compatible (c'est-à-dire que l'un est plus grand que l'autre au sens de la relation de généralisation) sont présents, on pourrait les fusionner en remplaçant leur marqueur par une pondération traduisant une imprécision sur le marqueur associé, par exemple une conjonction pondérée de marqueurs, où la définition du poids est également à étudier.

Exploitation de connaissances

Nous avons proposé l'algorithme *cgSpan*, présenté chapitre 5, ainsi qu'une étude comparative sur les modèles de graphes conceptuels flous, présentée chapitre 2.

Une première perspective est d'étendre ces travaux au cas des graphes conceptuels pondérés, c'est-à-dire de concevoir un algorithme d'extraction de motifs dans des graphes conceptuels pondérés. La nature ou la structure du motif en sortie serait à déterminer, notamment pour examiner s'il doit être pondéré ou non. Déterminer la définition de la mise en correspondance de deux graphes conceptuels imprécis serait le cœur du travail. Par exemple, cette mise en correspondance pourra être établie par la proposition d'une mesure de similarité entre deux graphes conceptuels, voire une opération d'agrégation pour la construction de motifs. Il sera en ce cas possible de profiter des apports existants en logique floue ou l'extraction de motifs flous par exemple.

Une seconde perspective est de proposer un algorithme d'extraction de motifs pondérés dans des graphes conceptuels non-pondérés. La sémantique de ces motifs et d'une relation d'homomorphisme spécifique sera alors à définir : par exemple, il serait possible de pondérer un motif en fonction de son support, ou de pondérer son support par rapport à la distance aux graphes qu'il recouvre. Cette distance peut correspondre à un homomorphisme prenant la généralisation en compte, l'imprécision, ou une combinaison de ces critères.

D'autre part, là où l'ajout de la taxonomie permet de généraliser des motifs pour recouvrir plus d'instances, sous certaines conditions, la définition d'un motif pondéré permettra de recouvrir également plus d'instances non-pondérées. Cela mène également à définir un opérateur combinant la généralisation et la *fuzzification*.

Enfin, au delà du cadre flou, des motifs de graphes conceptuels d'autres formes pourront être étudiés. Par exemple, le choix dans *cgSpan* de manipuler des graphes dont les

nœuds sont des briques élémentaires, c'est-à-dire une relation et son voisinage, peut être reformulé comme la formalisation d'un motif de graphe conceptuel imbriqué, où des connaissances sont représentées à différents niveaux.

Publications

Faci, A., Lesot, M.-J. & Laudy, C. (2021a). CG2A: Conceptual Graphs Generation Algorithm. *Int. Conf. of the European Society for Fuzzy Logic and Technology (EUSFLAT21)*.

Faci, A., Lesot, M.-J. & Laudy, C. (2021b). cgSpan: Pattern Mining in Conceptual Graphs. *Int. Conf. on Artificial Intelligence and Soft Computing (ICAISC21)*.

Faci, A., Lesot, M.-J. & Laudy, C. (2021c). Fuzzy Conceptual Graphs: a comparative discussion. *Symposium Series on Computational Intelligence, Foundation of Computational Intelligence (SSCI-FOCI21)*.

Bibliographie

Aggarwal, C. C. (2015). *Data mining: the textbook*. Springer.

Aggarwal, C. C., Bhuiyan, M. A. & Al Hasan, M. (2014). Frequent pattern mining algorithms: A survey. *Frequent pattern mining* (p. 19-64). Springer.

Agrawal, R. & Srikant, R. (1994). Fast algorithms for mining association rules. *Proc. of the 1994 Int. Conf. on Very Large Data Bases*, 487-499.

Agrawal, R., Imieliński, T. & Swami, A. (1993). Mining association rules between sets of items in large databases. *Proc. of the 1993 ACM SIGMOD Int. Conf. on Management of data*, 207-216.

Asaini. (2019). Implementation of apriori algorithm [https://github.com/asaini/Apriori]. https://github.com/asaini/Apriori

Baader, F., Horrocks, I., Lutz, C. & Sattler, U. (2017). An Introduction to Description Logic (C. U. Press, Éd.).

Baget, J.-F. (2003a). Simple conceptual graphs revisited: hypergraphs and conjunctive types for efficient projection algorithms. *Int. Conf. on Conceptual Structures*, 229-242.

Baget, J.-F. (2003b). Simple conceptual graphs revisited: hypergraphs and conjunctive types for efficient projection algorithms. *International Conference on Conceptual Structures*, 229-242.

Baget, J.-F. (2007). A datatype extension for simple conceptual graphs and conceptual graphs rules. *Int. Conf. on Conceptual Structures*, 83-96.

Baget, J.-F., Chein, M., Croitoru, M., Fortin, J., Genest, D., Gutierrez, A., Leclere, M., Mugnier, M.-L. & Salvat, E. (2009). RDF to conceptual graphs translations. *CS-TIW'09: 3rd Conceptual Structures Tool Interoperability Workshop@ ICCS'09: 17h Int. Conf. on Conceptual Structures*, (5662), 17.

Baget, J.-F., Chein, M., Croitoru, M., Gutierrez, A., Leclère, M. & Mugnier, M.-L. (2010a). Logical, graph based knowledge representation with CoGui. *GAOC: Graphes et Appariement d'Objets Complexes*, 15-25.

Baget, J.-F., Croitoru, M., Gutierrez, A., Leclère, M. & Mugnier, M.-L. (2010b). Translations between RDF(S) and Conceptual Graphs. In S. B. Heidelberg (Éd.), *Conceptual Structures: From Information to Intelligence* (p. 28-41). Springer.

Baget, J.-F. & Mugnier, M.-L. (2002). Extensions of simple conceptual graphs: the complexity of rules and constraints. *Journal of Artificial Intelligence Research*, *16*, 425-465.

Barwise, J. (1977). An introduction to first-order logic. *Studies in Logic and the Foundations of Mathematics* (p. 5-46). Elsevier.

Beierle, C., Hedtstueck, U., Pletat, U., Schmitt, P. H. & Siekmann, J. (1992). An order-sorted logic for knowledge representation systems. *Artificial intelligence*, *55*(2-3), 149-191.

Bouchon-Meunier, B. (1992). Linguistic hedges and fuzzy logic. *IEEE Int. Conf. on Fuzzy Systems*, 247-254.

Bouchon-Meunier, B. (2007). *La logique floue*. Que sais-je n° 2702.

Brickley, D., Guha, R. V. & McBride, B. (2014). RDF Schema 1.1. *W3C recommendation*, *25*.

Brin, S., Motwani, R. & Silverstein, C. (1997a). Beyond market baskets: Generalizing association rules to correlations. *Proc. of the 1997 ACM SIGMOD Int. Conf. on Management of data*, 265-276.

Brin, S., Motwani, R., Ullman, J. D. & Tsur, S. (1997b). Dynamic itemset counting and implication rules for market basket data. *Proc. of the 1997 ACM SIGMOD Int. Conf. on Management of data*, 255-264.

Buche, P., Dibie-Barthélemy, J. & Ibanescu, L. (2008). Ontology Mapping Using Fuzzy Conceptual Graphs and Rules. *ICCS Supplement*, *1724*, 17-24.

Buche, P., Fortin, J. & Gutierrez, A. (2014). Default reasoning implementation in CoGui. *Int. Conf. on Conceptual Structures*, 118-129.

Buche, P., Haemmerlé, O. & Thomopoulos, R. (2001). Representation of semi-structured imprecise data for fuzzy querying. *Proc. Joint 9th IFSA World Congress and 20th NAFIPS Int. Conf.*, *4*, 2126-2131.

Cakmak, A. & Ozsoyoglu, G. (2008). Taxonomy-superimposed graph mining. *Proc. of the 11th Int. Conf. on Extending database technology: Advances in database technology*, 217-228.

Cao, T. H. (1999). *Foundations of order-sorted fuzzy set logic programming in predicate logic and conceptual graphs* (thèse de doct.). University of Queensland.

Cao, T. H. & Creasy, P. N. (2000). Fuzzy types: a framework for handling uncertainty about types of objects. *Int. Journal of Approximate Reasoning*, *25*(3), 217-253. https://doi.org/10.1016/S0888-613X(00)00055-4

Cao, T. H., Creasy, P. N. & Wuwongse, V. (1997). Fuzzy types and their lattices. *Proc. of 6th Int. Conf. on Fuzzy Systems*, *2*, 805-812.

Cao, T. H. (2010). *Conceptual graphs and fuzzy logic: A fusion for representing and reasoning with linguistic information* (T. 306). Springer Science & Business Media.

Chein, M. & Mugnier, M.-L. (1997). Positive nested conceptual graphs. *Int. Conf. on Conceptual Structures*, 95-109.

Chein, M. & Mugnier, M.-L. (2004). Concept types and coreference in simple conceptual graphs. *Int. Conf. on Conceptual Structures*, 303-318.

Chein, M. & Mugnier, M.-L. (2008). *A Graph-Based Approach to Knowledge Representation: Computational Foundations of Conceptual Graphs*. Springer. https://doi.org/10.1007/978-1-84800-286-9

Chein, M. & Mugnier, M.-L. (2009). The BG family: facts, rules and constraints. *Graph-based Knowledge Representation: Computational Foundations of Conceptual Graphs*, 311-335.

Chein, M. & Mugnier, M.-L. (2014). Conceptual graphs are also graphs. *Int. Conf. on Conceptual Structures*, 1-18.

Chen, M.-C. (2007). Ranking discovered rules from data mining with multiple criteria by data envelopment analysis. *Expert Systems with Applications*, *33*(4), 1110-1116.

Chung, F. & Lu, L. (2002). The average distances in random graphs with given expected degrees. *Proceedings of the National Academy of Sciences*, *99*(25), 15879-15882.

Croitoru, M., Compatangelo, E. & Mellish, C. (2005). Hierarchical Knowledge Integration Using Layered Conceptual Graphs. In F. Dau, M.-L. Mugnier & G. Stumme (Éd.), *Conceptual Structures: Common Semantics for Sharing Knowledge* (p. 267-280). Springer Berlin Heidelberg.

Croitoru, M., Hu, B., Dashmapatra, S., Lewis, P., Dupplaw, D. & Xiao, L. (2007). A conceptual graph based approach to ontology similarity measure. *Int. Conf. on Conceptual Structures*, 154-164.

Dashti, Z., Pedram, M. M. & Shanbehzadeh, J. (2010). A multi-criteria decision making based method for ranking sequential patterns. *Proc. of the Int. MultiConf. of Enginners and Computer Scientists*, *1*.

De Runz, C., Giacometti, A., Markhoff, B. & Soulet, A. (2021). Découverte d'indicateurs de classement dans le Web des données. *Extraction et Gestion des Connaissances: Actes EGC'2021*, 35-46.

Dias, V., Teixeira, C. H., Guedes, D., Meira, W. & Parthasarathy, S. (2019). Fractal: A general-purpose graph pattern mining system. *Proc.of the 2019 Int. Conf. on Management of Data*, 1357-1374.

Dubois, D. & Prade, H. (1993). FUZZY NUMBERS: AN OVERVIEW. In D. Dubois, H. Prade & R. R. Yager (Éd.), *Readings in Fuzzy Sets for Intelligent Systems* (p. 112-148). Morgan Kaufmann. https://doi.org/https://doi.org/10.1016/B978-1-4832-1450-4.50015-8

Dubois, D. & Prade, H. (2012). *Possibility theory: an approach to computerized processing of uncertainty*. Springer Science & Business Media.

Dubois, D. & Prade, H. (2015). The legacy of 50 years of fuzzy sets: A discussion. *Fuzzy Sets and Systems*, *281*, 21-31.

Elseidy, M., Abdelhamid, E., Skiadopoulos, S. & Kalnis, P. (2014). Grami: Frequent subgraph and pattern mining in a single large graph. *Proc. of the VLDB Endowment*, *7*(7), 517-528.

Faci, A., Lesot, M.-J. & Laudy, C. (2021a). CG2A: Conceptual Graphs Generation Algorithm. *Int. Conf. of the European Society for Fuzzy Logic and Technology (EUSFLAT21)*.

Faci, A., Lesot, M.-J. & Laudy, C. (2021b). cgSpan: Pattern Mining in Conceptual Graphs. *Int. Conf. on Artificial Intelligence and Soft Computing (ICAISC21)*.

Faci, A., Lesot, M.-J. & Laudy, C. (2021c). Fuzzy Conceptual Graphs: a comparative discussion. *Symposium Series on Computational Intelligence, Foundation of Computational Intelligence (SSCI-FOCI21)*.

Ferreira, P. G. & Azevedo, P. J. (2005). Protein sequence pattern mining with constraints. *European Conference on Principles of Data Mining and Knowledge Discovery*, 96-107.

Fowler, M. D. (2019). John Cage's Silent Piece and the Japanese gardening technique of shakkei: Formalizing Whittington's conjecture through conceptual graphs. *Journal of Mathematics and Music*, *13*(1), 4-26.

Fu, Z., Huang, F., Ren, K., Weng, J. & Wang, C. (2017). Privacy-preserving smart semantic search based on conceptual graphs over encrypted outsourced data. *IEEE Transactions on Information Forensics and Security*, *12*(8), 1874-1884.

Ganascia, J.-G. & Velcin, J. (2004). Clustering of Conceptual Graphs with Sparse Data. *12th Int. Conf. on Conceptual Structures (ICCS)*, *3127*, 156-169. https://doi.org/10.1007/978-3-540-27769-9_10

Gkiokas, A. & Cristea, A. I. (2014). Training a Cognitive Agent to Acquire and Represent Knowledge from RSS feeds onto Conceptual Graphs. *IARIA COGNITIVE*, 184-194.

Goethals, B. (2003). *Survey on frequent pattern mining* (thèse de doct.). Univ. of Helsinki.

Gwadera, R. & Crestani, F. (2010). Ranking sequential patterns with respect to significance. *Pacific-Asia Conf. on Knowledge Discovery and Data Mining*, 286-299.

Han, J., Cheng, H., Xin, D. & Yan, X. (2007). Frequent pattern mining: current status and future directions. *Data mining and knowledge discovery*, *15*(1), 55-86.

Han, J., Pei, J. & Yin, Y. (2000). Mining frequent patterns without candidate generation. *ACM sigmod record*, *29*(2), 1-12.

Hell, P. & Nesetril, J. (2004). *Graphs and homomorphisms* (T. 28). OUP Oxford.

Inokuchi, A. (2004). Mining generalized substructures from a set of labeled graphs. *Fourth IEEE Int. Conf. on Data Mining (ICDM'04)*, 415-418.

Inokuchi, A., Washio, T. & Motoda, H. (2000). An apriori-based algorithm for mining frequent substructures from graph data. *European conference on principles of data mining and knowledge discovery*, 13-23.

Iyer, A. P., Liu, Z., Jin, X., Venkataraman, S., Braverman, V. & Stoica, I. (2018). ASAP: Fast, approximate graph pattern mining at scale. *Proc. of the 13th USENIX Symposium on Operating Systems Design and Implementation OSDI 18*, 745-761.

Jeavons, P., Cohen, D. & Cooper, M. C. (1998). Constraints, consistency and closure. *Artificial Intelligence*, *101*(1-2), 251-265.

Jin, Y. (2000). Fuzzy modeling of high-dimensional systems: complexity reduction and interpretability improvement. *IEEE Transactions on Fuzzy Systems*, *8*(2), 212-221.

Kamaruddin, S. S., Bakar, A. A., Hamdan, A. R. & Nor, F. M. (2008). Conceptual graph formalism for financial text representation. *2008 Int. Symp. on Information Technology*, *3*, 1-6.

Kamsu-Foguem, B., Diallo, G. & Foguem, C. (2013). Conceptual graph-based knowledge representation for supporting reasoning in African traditional medicine. *Engineering Applications of Artificial Intelligence*, *26*(4), 1348-1365.

Kamsu-Foguem, B., Tchuenté-Foguem, G. & Foguem, C. (2014). Using conceptual graphs for clinical guidelines representation and knowledge visualization. *Information Systems Frontiers*, *16*(4), 571-589.

Kaufmann, A. & Gupta, M. M. (1986). Introduction to fuzzy arithmetic : theory and applications.

Kowalski, P. & Martin, T. (2019). Embedding Uncertainty in Conceptual Graphs for Semantic Information Fusion. *2019 IEEE Conf. on Cognitive and Computational Aspects of Situation Management (CogSIMA)*, 1-8. https://doi.org/10.1109/COGSIMA.2019.8724358

Kuramochi, M. & Karypis, G. (2001). Frequent subgraph discovery. *Proc. 2001 IEEE Int. Conf. on data mining*, 313-320.

Laudy, C. & de Naurois, C. J. (2021). Nested Conceptual Graphs for Information Fusion Traceability. In T. Braun, M. Gehrke, T. Hanika & N. Hernandez (Éd.), *Graph-*

Based Representation and Reasoning - 26th Int. Conf. on Conceptual Structures (p. 19-33). Springer. https://doi.org/10.1007/978-3-030-86982-3_2

Laudy, C., Ganascia, J.-G. & Sedogbo, C. (2007). High-level fusion based on conceptual graphs. *Int. Conf. on Information Fusion*, 1-8.

Laudy, C., Habib, B. & Ganascia, J.-G. (2009). Fusion of Claude Bernard's Experiments for Scientific Discovery Reasoning. *17th International Conference on Conceptual Structures, ICCS 2009, 5662*, 219-232. https://doi.org/10.1007/978-3-642-03079-6_17

Lehmann, F. (1992). Semantic networks. *Computers & Mathematics with Applications*, *23*(2-5), 1-50.

Lesot, M.-J., Moyse, G. & Bouchon-Meunier, B. (2016). Interpretability of fuzzy linguistic summaries. *Fuzzy Sets and Systems*, *292*, 307-317. https://doi.org/https://doi.org/10.1016/j.fss.2014.10.019

Mannila, H., Toivonen, H. & Verkamo, A. (1994). Efficient algorithms for discovering association rules. *Proceeding of the AAAI'94 workshop knowledge discovery in databases*, 181-192.

Manola, F., Miller, E. & McBride, B. (2004). RDF primer. *W3C recommendation*, *10*(1-107).

Martin, T. & Azvine, B. (2017). Graded concepts and associations. *2017 IEEE Int. Conf. on Fuzzy Systems (FUZZ-IEEE)*, 1-6. https://doi.org/10.1109/FUZZ-IEEE.2017.8015748

McGuinness, D. L. & Van Harmelen, F. (2004). OWL web ontology language overview. *W3C recommendation*, *10*(10), 2004.

Micale, G., Giugno, R., Ferro, A., Mongiovi, M., Shasha, D. & Pulvirenti, A. (2018). Fast analytical methods for finding significant labeled graph motifs. *Data Mining and Knowledge Discovery*, *32*(2), 504-531.

Morton, S. (1987). *Conceptual graphs and fuzziness in artificial intelligence* (thèse de doct.). University of Bristol.

Mulhem, P., Leow, W. K. & Lee, Y. K. (2001). Fuzzy Conceptual Graphs for Matching Images of Natural Scenes. *Proc. of the 17th Int. Joint Conf. on Artificial Intelligence - Volume 2*, 1397-1402.

Naserasr, R., Rollová, E. & Sopena, É. (2015). Homomorphisms of signed graphs. *Journal of Graph Theory*, *79*(3), 178-212.

Pei, J., Han, J., Lu, H., Nishio, S., Tang, S. & Yang, D. (2001). H-mine: Hyper-structure mining of frequent patterns in large databases. *proceedings 2001 IEEE international conference on data mining*, 441-448.

Petermann, A. (s. d.). Implementation of DMGM-GSM algorithm. https://github.com/p3et

Petermann, A., Micale, G., Bergami, G., Pulvirenti, A. & Rahm, E. (2017). Mining and ranking of generalized multi-dimensional frequent subgraphs. *Proc. of the 12th Int. Conf. on Digital Information Management (ICDIM)*, 236-245.

Pinto, H., Han, J., Pei, J., Wang, K., Chen, Q. & Dayal, U. (2001). Multi-dimensional sequential pattern mining. *Proc. of the 10th Int. Conf. on Information and Knowledge Management*, 81-88.

Raimbault, T., Genest, D. & Loiseau, S. (2009). Conceptual Graphs and Datatypes. In S. Rudolph, F. Dau & S. O. Kuznetsov (Éd.), *Conceptual Structures: Leveraging Semantic Technologies* (p. 270-283). Springer Berlin Heidelberg.

Shah, N., Koutra, D., Zou, T., Gallagher, B. & Faloutsos, C. (2015). Timecrunch: Interpretable dynamic graph summarization. *Proc. of the 21th ACM SIGKDD Int. Conf. on knowledge discovery and data mining*, 1055-1064.

Sowa, J. F. (1983). Conceptual structures: information processing in mind and machine. *Computer Science*.

Sowa, J. F. (1987). Semantic networks. *NA*.

Sowa, J. F. (2008). Conceptual graphs. *Foundations of Artificial Intelligence*, *3*, 213-237.

Sowa, J. F. (2011). Peirce's tutorial on existential graphs.

Stern, D., Herbrich, R. & Graepel, T. (2006). Bayesian pattern ranking for move prediction in the game of Go. *Proc. of the 23rd Int. Conf. on Machine learning*, 873-880.

Thomopoulos, R., Buche, P. & Haemmerlé, O. (2003a). Different kinds of comparisons between fuzzy conceptual graphs. *Int. Conf. on Conceptual Structures*, 54-68.

Thomopoulos, R., Buche, P. & Haemmerlé, O. (2003b). Representation of weakly structured imprecise data for fuzzy querying. *Fuzzy Sets and Systems*, *140*(1), 111-128.

Thomopoulos, R., Buche, P. & Haemmerlé, O. (2006). Fuzzy sets defined on a hierarchical domain. *IEEE Transactions on knowledge and data engineering*, *18*(10), 1397-1410.

Tremblay, S., Gagnon, J.-F., Lafond, D., Hodgetts, H. M., Doiron, M. & Jeuniaux, P. P. (2017). A cognitive prosthesis for complex decision-making. *Applied ergonomics*, *58*, 349-360.

Vanetik, N., Gudes, E. & Shimony, S. E. (2002). Computing frequent graph patterns from semistructured data. *IEEE Int. Conf. on Data Mining*, 458-465.

Verboon, A. R. (2014). The medieval tree of Porphyry: An organic structure of logic. *The tree: Symbol, allegory, and mnemonic device in medieval art and thought* (p. 95-116).

Vlasenko, S. V., Efimenko, G. A., Gnezdilov, D. V. & Brikova, O. I. (2019). Approaches to Conceptual Graphs Notations Using in Digital Manufacturing Software Environments. *2019 IEEE Conf. of Russian Young Researchers in Electrical and Electronic*

Engineering (EIConRus), 731-735. https://doi.org/10.1109/EIConRus.2019.8656842

Wörlein, M., Meinl, T., Fischer, I. & Philippsen, M. (2005). A quantitative comparison of the subgraph miners MoFa, gSpan, FFSM, and Gaston. *European Conference on Principles of Data Mining and Knowledge Discovery*, 392-403.

Wuwongse, V. & Cao, T. H. (1996). Towards fuzzy conceptual graph programs. *Int. Conf. on Conceptual Structures*, 263-276.

Wuwongse, V. & Manzano, M. (1993). Fuzzy conceptual graphs. *Int. Conf. on Conceptual Structures*, 430-449.

Yan, X. & Han, J. (2002). gspan: Graph-based substructure pattern mining. *Proc. of IEEE Int. Conf. on Data Mining, ICDM'02*, 721-724.

Yin, Z., Cao, L., Han, J., Luo, J. & Huang, T. (2011). Diversified trajectory pattern ranking in geo-tagged social media. *Proc. of the 2011 SIAM Int. Conf. on Data Mining*, 980-991.

Zadeh, L. (1965). Fuzzy sets. *Information and Control*, *8*, 338-353.

Zadeh, L. A. (1975). The concept of a linguistic variable and its application to approximate reasoning-III. *Information sciences*, *9*(1), 43-80.