Un modèle de réification pour les graphes de propriétés : application à l'intégration de données et de connaissances multi-sources relatives aux handicaps

Selsebil Benelhaj-Sghaier, Annabelle Gillet, Éric Leclercq

Laboratoire d'Informatique de Bourgogne - EA 7534
Université de Bourgogne Franche-Comté
Dijon, France
Selsebil_Ben-El-Haj-Sghaier@etu.u-bourgogne.fr,{prenom}.{nom}@u-bourgogne.fr

RÉSUMÉ. *La diversité des données et des applications des systèmes d'information fait apparaître le besoin d'associer de la connaissance aux données afin de pouvoir leur ajouter une signification lors de leur manipulation. Les graphes de connaissances sont une solution flexible à ce besoin. Le modèle de graphe utilisé par les graphes de connaissances définit leur expressivité, c'est-à-dire les constructions qu'il est possible d'utiliser pour représenter la connaissance. Cependant, les modèles de graphe actuels possèdent des limites en terme d'expressivité, et ne permettent pas de représenter certaines relations complexes. Nous proposons d'étendre le modèle de graphe de propriétés afin d'améliorer son expressivité, en ajoutant la possibilité d'abstraire un sous-graphe sous la forme d'un nœud qui peut avoir des étiquettes, des propriétés et des liens vers d'autres nœuds, abstraits ou non. Cela permet de définir différents niveaux d'abstraction dans les graphes de connaissances et de leur associer des métadonnées.*

ABSTRACT. *The diversity of data and applications of information systems reveals the need to associate knowledge and data to add a signification when manipulating data. Knowledge graphs are a flexible solution to this need. The graph model used by a knowledge graph defines its expressivity, namely the constructions that can be used to represent knowledge. However, modern graph models have a limited expressivity, and cannot represent complex relationships. We propose to extend the property graph model to improve its expressivity, by adding the capability to abstract a subgraph as a node, that can have labels, properties and links toward other nodes, whether they are abstract or not. This mechanism allows to define multiple levels of abstraction with metadata in a knowledge graph.*

MOTS-CLÉS : *Graphe de connaissances, Graphe de propriétés, Abstraction multi-niveaux, Relations complexes.*

KEYWORDS: *Knowledge graph, Property graph, Multi-level abstraction, Complex relationships.*

1. Introduction

Les systèmes d'information doivent faire face à des données toujours de plus en plus volumineuses et variées, tout en mettant à disposition des utilisateurs une multitude d'applications se basant sur ces données. Chaque application, en fonction de ses objectifs, a besoin d'avoir accès à la signification des données qu'elle manipule, qui est alors apportée par des connaissances externes associées aux données.

Dans un contexte de données mutli-sources hétérogènes ayant recours à des connaissances variées pour supporter différentes applications, la représentation des connaissances sous forme d'ontologie a atteint ses limites à cause de sa restriction au point de vue d'un domaine spécifique sous forme de silo isolé, et l'alignement d'ontologies basée sur la concrétisation d'un consensus est difficile. Les graphes de connaissances (Ehrlinger, Wöß, 2016 ; Hogan *et al*., 2021 ; Gutiérrez, Sequeda, 2021) ont émergés comme une réponse flexible à ce besoin, en se concentrant sur l'intégration de données et connaissances hétérogènes (Li *et al*., 2021). Les graphes de connaissances ont également l'avantage de pouvoir utiliser différents modèles de graphe, tels que le graphe étiqueté ou le graphe de propriétés (Angles, 2018), et ainsi avoir accès à différents niveaux d'expressivité. Plus le modèle permet de réaliser des constructions variées à partir des éléments qui le composent, plus il est expressif.

Toutefois, l'expressivité des modèles de graphe actuels reste limitée, et certaines relations complexes, utiles lors de la modélisation des connaissances, sont difficilement représentables. Par exemple, on peut vouloir regrouper un sous-ensemble du graphe en un nouvel élément et lier cet élément au reste du graphe ou encore lui ajouter des métadonnées telles qu'une date de validité.

Dans cet article, nous proposons d'étendre le modèle de graphe de propriétés, en ajoutant un mécanisme d'abstraction s'inspirant de la réification RDF (Orlandi *et al*., 2021) afin d'améliorer son expressivité. De cette manière, il devient possible de représenter différents niveaux d'abstraction au sein d'un graphe de connaissance, d'ajouter des propriétés et des étiquettes aux niveaux d'abstraction et de définir des liens entre ces différents niveaux d'abstraction ou entre un élément du graphe et un niveau d'abstraction.

La suite de l'article est organisée de la manière suivante : la section 2 présente une étude de l'expressivité des différents modèles de graphes utilisés pour représenter des graphes de connaissances, la section 3 détaille les différentes approches existantes visant à améliorer l'expressivité des graphes de connaissances. Au niveau de la section 4, nous proposons une définition formelle de notre modèle. Dans la section 5, nous démontrons l'utilité de notre modèle sur des exemples dans le cadre du projet EASING qui a pour objectif de fournir des solutions d'hébergement à des personnes handicapées. Enfin, la section 6 conclut l'article et présente les orientations de nos travaux futurs.

2. Évolution de l'expressivité des modèles de graphes de connaissances

L'expressivité d'un graphe de connaissances dépend des opérateurs de construction supportés par le modèle de graphe qu'il utilise, permettant de décrire les entités et les relations. Dans cette section, nous explorerons l'expressivité de différents modèles des graphes.

En se basant sur la théorie des graphes, les modèles de graphe comportent un ensemble non vide de nœuds V représentant les entités et un ensemble d'arêtes E reliant des nœuds de V.

Le modèle du **graphe non orienté** est le modèle de graphe le plus basique en termes d'expressivité. Il est défini par le triplet (V, E), où $E \subseteq V \times V$. Ainsi, E représente les relations bidirectionnelles ou réciproques comme par exemple la relation "AMI_DE" mais il est incapable de représenter les relations unidirectionnelles comme par exemple la relation "PARENT_DE". Afin d'améliorer l'expressivité de ce modèle, il est nécessaire de rendre les arêtes directionnelles, comme il est possible de faire avec le modèle du graphe orienté.

Un **graphe orienté** est défini par le triplet (V, E, ρ), où $\rho : E \rightarrow V \times V$ est une fonction totale qui retourne le couple de nœuds associés par une arête. Par exemple, $\rho(e) = (v_1, v_2)$ indique que l'arête $e \in E$ est une arête orientée de v_1 à v_2. Cependant, s'il s'agit d'un multigraphe, c'est-à-dire s'il existe plusieurs arêtes entre les mêmes nœuds source et destination, il est difficile de différencier la signification de chaque arête. Afin d'améliorer l'expressivité du modèle de graphe orienté, le modèle du graphe orienté étiqueté peut être utilisé.

Un **graphe étiqueté** permet d'associer une étiquette à chaque nœud ou chaque arête. Un graphe étiqueté est défini comme un quadruplet (V, E, ρ, λ), où $\lambda : (E \cup V) \rightarrow \mathcal{L}$ est une fonction totale, avec $\mathcal{L}$ étant un ensemble d'étiquettes. Toutefois, cette représentation ne permet pas d'ajouter de métadonnées sur les liens ou les nœuds, contrairement au graphe de propriétés.

Un **graphe de propriétés** (Rodriguez, Neubauer, 2010 ; Angles, 2018) est défini comme un quintuplet $(V, E, \rho, \lambda, \sigma)$, où $\sigma : (V \cup E) \times Prop \rightarrow Val$ est une fonction partielle, dont $Prop$ est un ensemble fini des propriétés et Val est un ensemble de valeurs. Si $v \in V$ (resp., $e \in E$), $p \in Prop$, et $\sigma(v, p) = s$ (resp., $\sigma(e, p) = s$), alors s est la valeur de la propriété p du nœud v (resp., de l'arête e). Par rapport au modèle étiqueté, la fonction $\lambda : (V \cup E) \rightarrow 2^{\mathcal{L}}$, permet de définir un ensemble d'étiquettes pour chaque nœud et arête. Cependant, ce modèle de graphe, tout comme les modèles précédents, ne peut représenter que des relations binaires. En revanche, les hypergraphes dépassent cette limite et permettent de modéliser des relations n-aires.

Un **hypergraphe** (Berge, 1972) est une généralisation d'un graphe, capable de connecter plus de deux nœuds. Un hypergraphe $H = (V, E)$ est défini comme une famille d'hyper-arêtes E, où chaque hyper-arête est un sous-ensemble non vide de V, V étant un ensemble fini de nœuds. Le modèle d'hypergraphe a été utilisé pour repré-

senter les relations entre plusieurs entités, telles les nœuds de type diplôme, personne et université, liés par une hyper-arête représentant l'obtention du diplôme.

TABLEAU 1. *Tableau comparatif des capacités de représentation des différents modèles de graphe*

Modèle de graphe	Capacités de représentation						
	Arête orientée	Arête étiquetée	Nœud étiqueté	Propriété d'arête	Propriété du nœud	Relation binaire	Relation n-aire
Graphe non-orienté						✓	
Graphe orienté	✓					✓	
Graphe étiqueté	✓	✓	✓			✓	
Graphe de propriétés	✓	✓	✓	✓	✓	✓	
Hypergraphe						✓	✓

L'étude des modèles de graphe, synthétisée dans le tableau 1, montre d'une part que le modèle du graphe de propriétés généralise la plupart des modèles de graphe, et d'autre part que l'hypergraphe est le seul modèle qui puisse représenter les relations n-aires, sans toutefois permettre d'ajouter des étiquettes ou des propriétés. De plus, certains types de relations complexes, tels qu'un lien entre une sous partie d'un graphe et un de ses nœuds, ne peuvent être représentés par aucun de ces modèles. Dans la section suivante, nous explorons les travaux connexes qui ont essayé d'améliorer l'expressivité des modèles pour les graphes de connaissances.

3. Travaux connexes

RDF est un modèle pour échanger des données et des connaissances sur le Web. RDF est standardisé par le W3C et se base sur un modèle de graphe étiqueté. RDF représente des faits sous la forme de triplet (sujet, prédicat, objet) liant deux nœuds (le sujet et l'objet) par une arête orientée et étiquetée (le prédicat). Il s'agit de la seule construction du modèle. Cela implique que les faits et les propriétés d'objet sont représentés au même niveau, et qu'il est impossible de représenter des relations complexes. Par exemple, le fait "Le bâtiment 07 est accessible aux personnes aveugles depuis octobre 2023" pourrait être représenté en syntaxe Turtle avec deux triplets qui ne peuvent pas être liés :

```
:Building07 :AccessibleTo :Blind.
:Fact :since :October2023.
```

Par conséquent, il est nécessaire d'avoir un mécanisme qui exprime les métadonnées des triplets. La réification (Orlandi *et al.*, 2021) permet d'abstraire un triplet RDF pour

lui ajouter des métadonnées. Quatre types de réification RDF ont été proposés : la réification standard, la réification via les graphes nommés, la réification via la propriété singleton et la réification RDF*.

La **réification standard** (Manola, Miller, 2004), connue sous le terme RDF Primer, fait référence au processus de représentation d'un triplet RDF en tant que nouvelle ressource, qui est une instance de la classe `Statement`, avec les principales propriétés `rdf:subject`, `rdf:predicate` et `rdf:object`. Avec la réification standard, le fait "Le bâtiment 07 est accessible aux personnes aveugles depuis octobre 2023" peut être représenté comme suit:

```
_:x rdf:type rdf:Statement.
_:x rdf:subject :Building07.
_:x rdf:predicate :AccessibleTo.
_:x rdf:object :Blind.
_:x  :since :October2023.
```

Il est important de noter que la nouvelle ressource de type `rdf:Statement` est représentée, dans un graphe RDF, sous la forme d'un nœud anonyme (ou encore *blank node*), sans IRI. Chen *et al*. (2012) ont expliqué l'utilité des *blank nodes* dans un graphe RDF notamment pour relier un nœud abstrait à des métadonnées.

La **réification basée sur les graphes nommés** (Carroll *et al*., 2005) définit un graphe nommé G comme un sous-graphe associé à un identifiant unique. Il est représenté sous la forme d'une paire (G, ID), où ID est l'identifiant du graphe nommé G. La notion de graphe nommé a été par la suite standardisée par le W3C (Ivan, 2010). Cette approche de réification utilise le graphe nommé comme un contexte dans lequel les triplets RDF réifiés sont regroupés. Ce mécanisme peut être utilisé pour associer un triplet principal (comme `:Building07 :AccessibleTo :Blind.`) et les triplets qui le décrivent (comme `:Fact :since :October2023.`). L'exemple précédent peut être représenté comme suit :

```
:Building07Info {
:Building07 :AccessibleTo :Blind.
:Fact :since :October2023.}
```

En se basant sur l'idée des graphes nommés, Stoermer *et al*. (2006) ont proposé d'améliorer l'expressivité du graphe RDF en permettant d'associer à un triplet un ou plusieurs contextes représentés au moyen de graphes nommés. Les auteurs ont également modélisé les relations entre les contextes par le triplet (c_x, R, c_y) où c_x et c_y sont les IRI de deux graphes nommés représentant deux contextes différents et R est l'IRI de la relation entre eux. D'autres approches se sont intéressées à l'abstraction des graphes orientés étiquetés. Poulovassilis et Levene (1994) ont proposé l'*hypernode graph*, qui permet de représenter un sous-graphe d'un graphe orienté étiqueté sous forme d'un nœud. Les sous-graphes étant des nœuds ils peuvent alors être reliés entre eux. Les *nested graphs* (Chein *et al*., 1998) proposent également un mécanisme d'abstraction, dans lequel un sous-graphe correspondant à un certain contexte peut être représenté par un nœud pouvant être lié aux autres nœuds du graphe.

La **réification de propriété singleton** (Nguyen *et al*., 2014) ajoute un prédicat unique au prédicat original d'un triplet RDF auquel il est possible d'ajouter des métadonnées grâce à des triplets supplémentaires liés au prédicat unique considéré en tant que sujet. Il s'agit d'une approche alternative à la réification standard principalement basée sur les prédicats. Par exemple, en utilisant la propriété singleton, le fait précédent devient :

```
:Building07 :AccessibleTo#1 :Blind.
:AccessibleTo#1 rdf:singletonPropertyOf :AccessibleTo.
:AccessibleTo#1 :since :October2023.
```

Avec la réification par propriété singleton, le prédicat `AccessibleTo` du triplet original est étendu à un prédicat unique `AccessibleTo#1` qui est ensuite utilisé comme sujet pour ajouter des faits supplémentaires. Les travaux de Rosso *et al*. (2020) se sont inspirés de cette technique d'ajout des métadonnées sur les arêtes en proposant d'accorder à chaque arête le couple *(qualifier, value)* représentant le nom de la métadonnée et sa valeur. Ce modèle se rapproche alors de celui du graphe de propriétés.

La **réification RDF*** (Hartig, 2017) est une simplification syntaxique visant à simplifier la réification standard en évitant de créer de nouvelles ressources. Selon RDF*, notre exemple précédent est encodé à l'aide d'un seul triplet RDF* :

```
<<:Building07 :AccessibleTo :Blind>> :since :October2023.
```

Ce triplet RDF* imbriqué est appelé le triplet de métadonnées dont le sujet est encadré par « et ». Kovacevic *et al*. (2022) se sont basés sur la technique RDF* pour annoter les triplets de données avec des métadonnées temporelles. Ils proposent d'ajouter deux attributs au triplet réifié : `valid_from`, désignant la date de création du `timestamp` et `valid_until` désignant la date d'expiration, les deux étant liés à un *timestamp*. Xiong *et al*. (2023) ont proposé les *nested facts*, consistant à ajouter des liens entre des liens. Cette approche peut être vue comme une réification RDF* entre deux triplets.

Après avoir étudié les travaux de la littérature, on peut distinguer deux familles d'approches distinctes visant à améliorer l'expressivité des modèles pour les graphes de connaissances. En premier lieu, il existe des approches qui se rapprochent du modèle graphe de propriétés, qui permet d'englober une partie des méthodes de réification grâce aux propriétés des arêtes. En deuxième lieu, on trouve des approches qui s'apparentent aux techniques de la reification RDF, que l'on peut diviser en deux catégories : 1) la manipulation de triplets, comprenant la réification standard, RDF*, et celle de la propriété singleton, et 2) le regroupement par sous-graphe, représentée par le graphe nommé. Chaque catégorie présente ses propres limites : la catégorie de manipulation de triplets ne peut réifier qu'un seul triplet RDF (soit deux nœuds connectés), tandis que la catégorie de sous-graphe ne dispose pas d'un mécanisme d'ajout de métadonnées sur le sous-graphe lui même. Certaines approches permettent de créer des liens

entre des nœuds pouvant représenter un sous-graphe abstrait, mais ne permettent toutefois pas de relier un élément du sous-graphe avec le reste du graphe. De cette étude, deux constats peuvent être faits : 1) le modèle de graphe de propriétés permet à la fois d'abstraire la plupart des modèles de graphe ainsi que plusieurs techniques de réification, et 2) les relations complexes ne peuvent pas facilement être représentées.

4. Approche proposée

Pour améliorer l'expressivité du graphe de propriétés, nous nous inspirons du principe de réification afin de proposer un modèle permettant de représenter un sous-graphe (un ensemble de nœuds, arêtes, propriétés et étiquettes) sous la forme d'un nœud, agissant lui-même comme un nœud standard du graphe. Ce mécanisme peut éventuellement être récursif et ainsi permettre de modéliser différents niveaux d'abstraction. Dans la suite de la section nous expliquons les principes du modèle puis nous proposons une définition formelle.

4.1. Principes fondamentaux du modèle proposé

Tout d'abord, les propriétés de données et les propriétés d'objets sont toutes deux représentées par une arête dans RDF, par contre, dans un graphe de propriétés, une propriété de données devient une propriété d'un nœud. Par conséquent, les propriétés de nœuds doivent être considérées pour conserver au moins le même niveau d'expressivité que les techniques de réification manipulant les triplets. Les propriétés d'arêtes, quant à elles, doivent aussi être considérées pour exploiter au maximum le modèle du graphe de propriétés. De plus, dans un graphe de propriétés, chaque nœud et arête peut avoir plusieurs étiquettes, plutôt qu'une seule comme c'est le cas dans un graphe RDF. Il faut donc pouvoir définir un sous-ensemble des étiquettes pour un nœud ou une arête. Les relations complexes, entre des sous-graphes, doivent également pouvoir être représentées afin d'augmenter l'expressivité du modèle.

4.2. Définition formelle du modèle proposé

À partir des principes définis dans la sous-section précédente, nous proposons un modèle pouvant représenter un ensemble de nœuds et potentiellement d'arêtes (les nœuds peuvent être partiellement/totalement connectés ou même déconnectés) ainsi que certaines de leurs étiquettes et propriétés pour obtenir un sous-graphe. Ce sous-graphe prend la forme d'une nœud abstrait, pouvant avoir des étiquettes, des propriétés et des liens vers d'autres nœuds standards ou abstraits.

Le modèle proposé étend le modèle de graphe de propriétés de la manière suivante :

$$G = (V, E, R, \rho, \lambda, \sigma, \alpha)$$

où :

– $R \subseteq V$, R est un sous-ensemble de V contenant les nœuds réifiés;

– $\alpha : R \rightarrow SG$ est une fonction totale qui fait correspondre les nœuds réifiés à leur sous-graphe. Si $r \in R$ et $sg \in SG$, alors $\alpha(r) = sg$ où sg est le sous-graphe du nœud réifié r. SG est défini par un ensemble fini de tuples sg de la forme suivante :

$$sg = (V_r, E_r, R_r, \rho_r, \lambda_r, \sigma_r, \alpha_r)$$

où : $V_r \subseteq V$, $E_r \subseteq E$, $R_r \subseteq R$ avec les fonctions :

- $\rho_r = \rho|_{E_r}$
- $\lambda_r = \lambda|_{V_r \cup E_r}$ c'est-à-dire $\lambda_r : (V_r \cup E_r) \rightarrow 2^{\mathcal{L}_r}$
- $\sigma_r = \sigma|_{(V_r \cup E_r) \times Prop_r}$
- $\alpha_r = \alpha|_{R_r}$

Les fonctions $\rho_r, \lambda_r, \sigma_r$ et α_r sont les restrictions des fonctions ρ, λ, σ et α qui associent respectivement à une arête orientée ses nœuds, à un nœud ou une arrête un ensemble d'étiquettes, à un nœud ou une arrête ses propriétés et leur valeur, et à un nœud réifié son sous-graphe. $Prop_r$ est un ensemble fini de propriétés sélectionnées du sous-graphe et $\mathcal{L}_r$ est un ensemble fini d'étiquettes sélectionnées, donc $Prop_r \subseteq Prop$ et $\mathcal{L}_r \subseteq \mathcal{L}$.

Le modèle ainsi défini permet alors les constructions suivantes :

– **l'abstraction d'un sous-graphe sous forme d'un nœud réifié** au moyen de la fonction σ. Un nœud réifié se compose alors d'un ensemble de nœuds, d'arêtes ainsi que de certaines de leurs étiquettes et propriétés ;

– **l'abstraction récursive** car $R \subset V$. Ainsi, un nœud réifié peut contenir d'autres nœuds réifiés et définir de cette manière des niveaux d'abstraction ;

– **l'ajout de métadonnées sur un sous-graphe** puisqu'un nœud réifié est un élément de V, donc il peut avoir à la fois des étiquettes et des propriétés. Il peut également avoir des liens avec d'autres nœuds standards ou réifiés du graphe.

4.3. Spécification du nœud réifié

Pour construire un nœud réifié, il faut tout d'abord spécifier les éléments concernés par la réification. Comme indiqué dans la définition du modèle, il est possible de réifier les nœuds, les arêtes, ainsi que certaines de leurs étiquettes et propriétés. Donc, la fonction de construction du nœud réifié doit accepter en paramètres ces éléments individuellement ou toute combinaison d'entre eux. Pour ce faire, nous proposons la fonction suivante :

$$\beta(V_r, E_r, \lambda_r, \sigma_r) = r$$

où:

– V_r est l'ensemble des nœuds sélectionnés

– E_r est l'ensemble des arêtes sélectionnées

- $\lambda_r : (V_r \cup E_r) \rightarrow 2^{\mathcal{L}_r}$ se restreint sur les étiquettes sélectionnées
- $\sigma_r : (V_r \cup E_r) \times Prop_r \rightarrow Val$ se restreint sur les propriétés sélectionnées
- $\rho_r = \rho|_{E_r}$ est une restriction de la fonction ρ à E_r

Pour obtenir R_r tel qu'il est indiqué dans le modèle, nous nous appuyons sur R du graphe global, car les nœuds réifiés d'un sous-graphe sont également des nœuds réifiés dans le graphe contenant le sous-graphe. Par conséquent, $R_r = \{v | v \in R \cap V_r\}$.

Étant donné que λ_r et σ_r ont leurs domaines restreints selon les nœuds et les arêtes sélectionnés, cela garantit que les étiquettes et propriétés sélectionnées appartiennent à un nœud ou une arête existant dans le sous-graphe de la réification.

5. Validation préliminaire

Dans cette section, nous identifions des cas d'utilisation réels qui illustrent la nécessité de représenter des relations complexes dans un graphe de connaissances. Nous montrons comment notre approche répond aux exigences de chaque cas d'utilisation.

Les cas d'utilisation abordés dans cette section sont dérivés de l'un des *Work Packages* du projet EASING (mobility of pErsons And acceSsible housINg). Il s'agit d'un projet français visant à aider les personnes handicapées à trouver un logement adapté à leurs besoins. Nous nous concentrons sur la vérification de la conformité des bâtiments par rapport au Code de la Construction et d'Habitation (CCH) régissant l'accessibilité pour les personnes handicapées.

Les éléments du bâtiment sont représentées à l'aide de l'ontologie IfcOWL [1] utilisée comme un schéma du graphe de connaissances utilisant le modèle du graphe de propriétés. Pour le CCH, nous nous référons aux articles du décret français du 20 avril 2017 [2] régissant l'accessibilité des établissement publics pour les personnes handicapées.

Certains termes du CCH ne correspondent pas directement aux entités du schéma du graphe de connaissances qui représente le bâtiment. Par conséquent, la mise en place d'un mécanisme de vérification de la conformité des bâtiments nécessite de représenter les termes du code de construction en fonction des éléments du bâtiment.

Nous utilisons trois cas d'utilisation différents pour montrer les exigences particulières en terme de représentation des connaissances et pour démontrer comment notre modèle peut fournir une solution appropriée. Au niveau du 1er cas d'utilisation "Représentation du terme : circulation horizontale", nous appliquons la réification sur des nœuds connectés avec leurs étiquettes et propriétés. Pour le 2e cas d'utilisation "Représentation du terme : cheminement extérieur" nous appliquons la réification récursive dont un nœud réifié contient un autre nœud réifié. Pour le 3e cas d'utilisation "Repré-

1. https://github.com/buildingSMART/ifcOWL/blob/master/IFC2X3_Final.owl
2. https://www.legifrance.gouv.fr/loda/id/JORFTEXT000034485459/

sentation du terme : circulation verticale" nous appliquons la réification des nœuds non reliés ainsi que de leurs étiquettes en ajoutant un lien entre des nœuds réifiés.

5.1. Représentation du terme "Circulation horizontale"

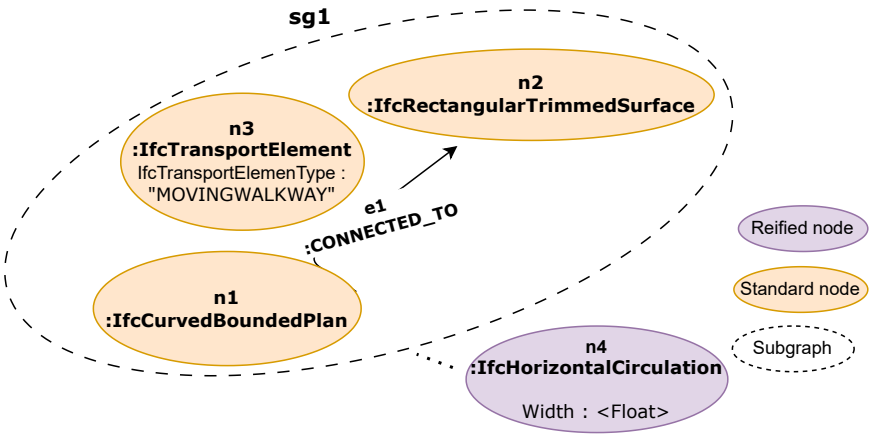

FIGURE 1. *La représentation du terme de circulation horizontale en utilisant la réification*

Le sixième article du CCH indique : "Les circulations horizontales doivent avoir une largeur minimale de 1,40 mètre". Le terme utilisé dans le CCH est "circulation horizontale". Dans la représentation IfcOWL, les éléments du bâtiment constituant ce terme sont les suivants : un plan borné courbé (IfcCurvedBoundedPlan) connecté à une surface rectangulaire rognée (IfcRectangularTrimmedSurface), et un élément de transport (IfcTransportElement) qui doit être une passerelle. Nous devons réifier les éléments ci-dessus en un nœud abstrait étiqueté IfcHorizontalCirculation.

Pour ce faire, nous créons un nœud réifié IfcHorizontalCirculation via la réification des trois nœuds n_1, n_2, n_3, de l'arête e_1, des étiquettes "IfcCurvedBoundedPlan" de n_1,"IfcRectangularTrimmedSurface" de n_2, "IfcTransportElement" de n_3 et de l'étiquette "CONNECTED_TO" de e_1, ainsi que de la propriété "IfcTransportElementType" égale à "MOVINGWALKWAY" de n_3 (voir la figure 1). En appliquant la définition de notre modèle, le sous-graphe sg_1 qui constitue le nœud réifié IfcHorizontalCirculation est le suivant:

$$sg_1 = (V_r, E_r, R_r, \rho_r, \lambda_r, \sigma_r, \alpha_r)$$

où:

– $V_r = \{n_1, n_2, n_3\}$, $E_r = \{e_1\}$, $R_r = \emptyset$

- $\rho_r(e_1) = (n_1, n_2)$
- $\lambda_r(n_1) = "IfcCurvedBoundedPlan"$
- $\lambda_r(n_2) = "IfcRectangularTrimmedSurface"$
- $\lambda_r(n_3) = "IfcTransportElement"$
- $\lambda_r(e_1) = "CONNECTED\_TO"$
- $\sigma_r(n_3, "IfcTransportElementType") = "MOVINGWALKWAY"$

$$\beta(V_r, E_r, \lambda_r, \sigma_r) = n_4$$

5.2. Représentation du terme "Cheminement extérieur"

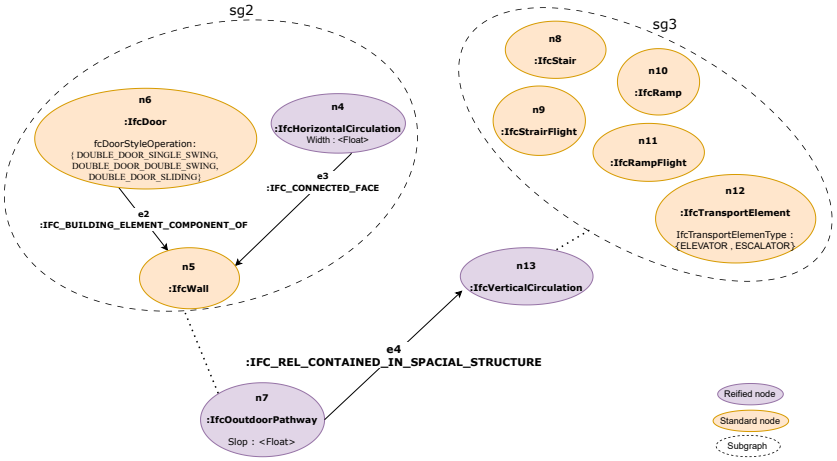

FIGURE 2. *Représentation des termes Cheminement Extérieur et Circulation Verticale en utilisant la réification*

Le deuxième article du CCH indique : "Si un cheminement extérieur a une pente de 5% ou moins, il doit disposer d'un équipement de circulation verticale". Le premier terme du code est "Cheminement Extérieur" ou encore *Outdoor Pathway*. Un chemin extérieur est un cheminement horizontal se terminant par une porte. Ainsi, les nœuds constituant ce terme sont étiquetés IfcDoor et IfcHorizontalCirculation. Pour la représentation du terme "Cheminement Extérieur", le nœud réifié IfcHorizontalCirculation qui a été précédemment créé servira à représenter le terme "Cheminement Extérieur".

Ainsi, nous sélectionnons le nœud réifié n_4 avec son étiquette IfcHorizontalCirculation, le nœud standard n_5 avec son étiquette IfcWall, le nœud standard n_6 avec son étiquette IfcDoor avec sa propriété IfcDoorStyleOperation, l'arête e_2 avec son étiquette IFC_BUILDING_ELEMENT_COMPONENT_OF et l'arête e_3 avec son étiquette IFC_CONNECTED_FACE (voir la partie gauche de la figure 2).

Nous pouvons noter que le nœud réifié IfcOutdoorPathway contient un autre nœud réifié IfcHorizontalCirculation. Il s'agit d'une réification récursive qui permet de représenter plusieurs niveaux d'abstraction. En appliquant la définition de notre modèle, le sous-graphe sg_2 qui constitue le nœud réifié IfcOutdoorPathway est le suivant:

$$sg_2 = (V_r, E_r, R_r, \rho_r, \lambda_r, \sigma_r, \alpha_r)$$

où:

- $V_r = \{n_4, n_5, n_6\}$, $E_r = \{e_2, e_3\}$, $R_r = \{n_4\}$
- $\rho_r(e_2) = (n_5, n_6)$
- $\rho_r(e_3) = (n_4, n_5)$
- $\lambda_r(n_4) = "IfcHorizontalCirculation"$
- $\lambda_r(n_5) = "IfcWall"$
- $\lambda_r(n_6) = "IfcDoor"$
- $\lambda_r(e_2) = "IFC\_BUILDING\_ELEMENT\_COMPONENT\_OF"$
- $\lambda_r(e_3) = "IFC\_CONNECTED\_FACE"$
- $\sigma_r(n_6, "IfcDoorStyleOperation") = \{"DOUBLE\_DOOR\_SINGLE\_SWING", "DOUBLE\_DOOR\_DOUBLE\_SWING", "DOUBLE\_DOOR\_SLIDING"\}$
- $\alpha_r(n_4) = sg_1$

$$\beta(V_r, E_r, \lambda_r, \sigma_r) = n_7$$

5.3. Représentation du terme "Circulation Verticale"

Dans son deuxième, septième et seizième article, le CCH utilise le terme "Circulation Verticale". Les nœuds constituant ce terme sont : IfcStair, IfcStairFlight, IfcRamp, IfcRampFlight et IfcTransportElement. Le terme "Circulation Verticale" ou *Vertical Circulation* représente des nœuds qui sont totalement déconnectés (aucune arête entre eux).

Ainsi, pour créer un nœud réifié étiqueté IfcVerticalCirculation, nous sélectionnons les nœuds n_8, n_9, n_{10} et n_{11} avec leur étiquette respective, ainsi que le nœud n_{12} avec son étiquette IfcTransportElement et sa propriété IfcTransportElementType (voir la partie droite de la figure 2). En appliquant la définition de notre modèle, le sous-graphe sg_3 qui constitue le nœud réifié IfcVerticalCirculation est le suivant:

$$sg_3 = (V_r, E_r, R_r, \rho_r, \lambda_r, \sigma_r, \alpha_r)$$

où:

- $V_r = \{n_8, n_9, n_{10}, n_{11}, n_{12}\}$, $E_r = \emptyset$, $R_r = \emptyset$
- $\lambda_r(n_8) = "IfcStair"$
- $\lambda_r(n_9) = "IfcStairFlight"$
- $\lambda_r(n_{10}) = "IfcRamp"$

- $\lambda_r(n_{11}) = "IfcRampFlight"$
- $\lambda_r(n_{12}) = "IfcTransportElement"$
- $\sigma_r(n_{12}, "IfcTransportElementType") = \{"ELEVATOR", "ESCALATOR"\}$

$$\beta(V_r, E_r, \lambda_r, \sigma_r) = n_{13}$$

À travers les cas d'utilisation traités ci-dessus, nous avons démontré comment notre approche a contribué à améliorer l'expressivité du graphe de propriétés en représentant des relations complexes à l'aide de l'abstraction des nœuds, des arêtes, de leurs propriétés et étiquettes ou bien en ajoutant des niveaux d'abstraction avec réification récursive.

6. Conclusion et perspectives

Afin de pouvoir augmenter l'expressivité des graphes de connaissances et représenter des relations complexes, nous avons proposé un modèle basé sur le graphe de propriétés qui permet d'abstraire un sous-graphe sous forme de nœud se comportant comme un nœud standard du graphe, afin de pouvoir lui ajouter des liens, des propriétés et des étiquettes. Grâce à la récursivité de la définition, il devient alors possible de définir plusieurs niveaux d'abstraction avec leurs métadonnées. Nous avons également illustré l'utilité du modèle proposé à travers des exemples concrets dans le cadre du projet EASING.

Nos travaux futurs se concentrent sur deux axes principaux. Premièrement, nous prévoyons de passer à l'implémentation technique de notre approche. Deuxièmement, nous prévoyons d'adapter les langages de requête de graphe de propriétés actuels à notre modèle. Cette adaptation est essentielle pour permettre une gestion efficace de l'abstraction à plusieurs niveaux lors de l'interrogation des graphes de connaissances. Par exemple, les requêtes devront être capables d'assurer l'agrégation des propriétés, notamment lorsqu'il s'agit de calculer une propriété à partir des propriétés des éléments constitutifs d'un nœud abstrait. Ces deux axes représentent une étape cruciale pour exploiter pleinement le potentiel de notre modèle dans des contextes pratiques et pour faciliter son intégration dans les applications et les systèmes informatiques existants.

Bibliographie

Angles R. (2018). The property graph database model. In *Amw*.

Berge C. (1972). Graphes et hypergraphes. *Dunod*.

Carroll J. J., Bizer C., Hayes P., Stickler P. (2005). Named graphs. *Journal of Web Semantics*, vol. 3, nº 4, p. 247–267.

Chein M., Mugnier M.-L., Simonet G. (1998). Nested graphs: A graph-based knowledge representation model with fol semantics. In *Kr*, p. 524–535.

Chen L., Zhang H., Chen Y., Guo W. (2012). Blank nodes in rdf. *J. Softw.*, vol. 7, nº 9, p. 1993–1999.

Ehrlinger L., Wöß W. (2016). Towards a definition of knowledge graphs. *SEMANTiCS (Posters, Demos, SuCCESS)*, vol. 48, nº 1-4, p. 2.

Gutiérrez C., Sequeda J. F. (2021). Knowledge graphs. *Communications of the ACM*, vol. 64, nº 3, p. 96–104.

Hartig O. (2017). Foundations of RDF* and SPARQL*:(an alternative approach to statement-level metadata in RDF). In *Amw 2017 11th alberto mendelzon international workshop on foundations of data management and the web, montevideo, uruguay, june 7-9, 2017.*, vol. 1912.

Hogan A., Blomqvist E., Cochez M., d'Amato C., Melo G. D., Gutierrez C. *et al.* (2021). Knowledge graphs. *ACM Computing Surveys (CSUR)*, vol. 54, nº 4, p. 1–37.

Ivan H. (2010). *Rdf graph literals and named graphs.* Consulté sur https://www.w3.org/2009/07/NamedGraph.html

Kovacevic F., Ekaputra F. J., Miksa T., Rauber A. (2022). Starvers-versioning and timestamping rdf data by means of rdf*-an approach based on annotated triples.

Li X., Lyu M., Wang Z., Chen C.-H., Zheng P. (2021). Exploiting knowledge graphs in industrial products and services: a survey of key aspects, challenges, and future perspectives. *Computers in Industry*, vol. 129, p. 103449.

Manola F., Miller E. (2004). *Rdf reification.* Consulté sur {https://www.w3.org/TR/rdf-primer/\#reification}

Nguyen V., Bodenreider O., Sheth A. (2014). Don't like RDF reification? making statements about statements using singleton property. In *Proceedings of the 23rd international conference on world wide web*, p. 759–770.

Orlandi F., Graux D., O'Sullivan D. (2021). Benchmarking RDF metadata representations: Reification, singleton property and RDF. In *2021 IEEE 15th International Conference on Semantic Computing (ICSC)*, p. 233–240.

Poulovassilis A., Levene M. (1994). A nested-graph model for the representation and manipulation of complex objects. *ACM Transactions on Information Systems (TOIS)*, vol. 12, nº 1, p. 35–68.

Rodriguez M. A., Neubauer P. (2010). Constructions from dots and lines. *arXiv preprint arXiv:1006.2361*.

Rosso P., Yang D., Cudré-Mauroux P. (2020). Beyond triplets: hyper-relational knowledge graph embedding for link prediction. In *Proceedings of the web conference 2020*, p. 1885–1896.

Stoermer H., Palmisano I., Redavid D., Iannone L., Bouquet P., Semeraro G. *et al.* (2006). rdf and contexts: Use of sparql and named graphs to achieve contextualization. In *Proc. of 2006 jena user conference.*

Xiong B., Nayyeri M., Luo L., Wang Z., Pan S., Staab S. (2023). Neste: Modeling nested relational structures for knowledge graph reasoning. *arXiv preprint arXiv:2312.09219*.