Evolution d'une RTO dédiée à la représentation de relations n-aires

Rim Touhami, Patrice Buche, Juliette Dibie-Barthelemy, Liliana Ibanescu

Évolution d'une RTO dédiée à la représentation de relations n-aires

Résumé. Nous nous intéressons dans cet article à la problématique d'évolution d'une Ressource Termino-Ontologique (RTO) permettant de représenter des relations n-aires. Tout au long de son cycle de vie, une RTO est amenée à évoluer pour diverses raisons (e.g. prise en compte de nouvelles données, de nouveaux besoins utilisateurs, alignement avec de nouvelles ontologies). Nous présentons dans cet article la représentation formelle des changements applicables à notre RTO permettant de modifier sa structure tout en maintenant sa cohérence structurelle. Nous illustrerons nos propos sur une RTO dédiée à la représentation de relations n-aires entre des données expérimentales quantitatives dans le domaine du risque alimentaire microbiologique étendu aux emballages.

1 Introduction

Le Web d'aujourd'hui est une source inépuisable d'informations, et permet, entre autres, d'alimenter des entrepôts de données. Afin d'intégrer des données dans un entrepôt, une première étape consiste à harmoniser ces données avec les données locales de l'entrepôt, c'est à dire que les données du Web doivent s'exprimer dans le même vocabulaire, généralement représenté à l'aide d'une ontologie, que celui utilisé par les données locales. La notion d'ontologie est une notion clé du Web Sémantique et des recherches en intégration de données. Afin d'établir une distinction claire entre la composante terminologique et la composante conceptuelle, de travaux de McCrae et al. (2011); Reymonet et al. (2009); Roche et al. (2009) ont proposé d'associer une partie terminologique et/ou linguistique aux ontologies. Il serait illusoire de croire que les ontologies commes les ressources termino-ontologiques peuvent rester stables dans ce monde en perpétuelles mutations, il est donc nécessaire d'étudier leurs évolutions pour répondre à différents besoins de changements (e.g. apparition de nouvelles connaissances, de nouveaux concepts ou de nouveaux termes).

Dans cet article, nous nous intéressons à la problématique d'évolution d'une Ressource Termino-Ontologique (RTO) dédiée à la représentation de relations n-aires. D'après Noy et

Rector (2006), une relation n-aire est une relation qui est définie entre au moins deux arguments. Nous avons choisi de représenter des relations n-aires sans arguments différenciés telles que définies par le W3C [1], ce qui correspond au cas le plus général d'utilisation des relations n-aires, le cas 3. Nous avons de plus choisi d'utiliser le "patron 1" qui consiste à représenter une relation n-aire à l'aide d'un concept, relié à ses arguments par des propriétés (cf. figure 1).

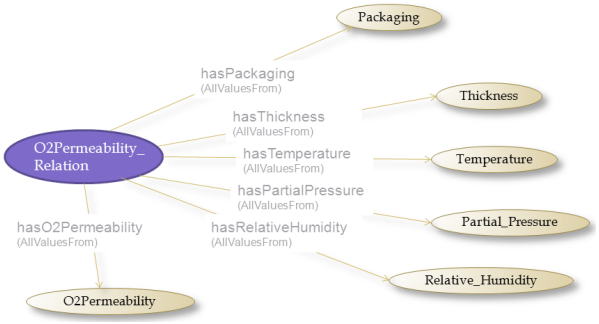

FIG. 1 – *Exemple d'une relation n-aire dans le domaine de la microbiologie.*

Supposons que l'on veuille ajouter la relation n-aire O2Permeability_Relation de la figure 1 à notre RTO, l'existence d'interdépendances entre la relation et ses six arguments nécessite d'effectuer un nombre important de changements dans la RTO pour y parvenir (e.g. ajout de concepts, de propriétés, de restrictions, de termes). En outre, ces changements peuvent engendrer des incohérences dans la RTO. Nous présentons ici la représentation formelle des changements applicables à notre RTO dédiée à la représentation de relations n-aires permettant de modifier sa structure tout en maintenant sa cohérence structurelle. Nous nous intéressons plus particulièrement aux relations n-aires entre des données quantitatives expérimentales, ce qui suppose d'apporter une attention particulière à la gestion des arguments numériques (i.e. les quantités) et leurs unités de mesure. Dans la section 2, nous présentons la modélisation de la RTO dédiée à la représentation de relations n-aires et la notion de RTO structurellement cohérente. Les changements élémentaires et composés permettant de faire évoluer la RTO sont détaillés dans la section 3. Notre approche est comparée à l'état de l'art dans la section 4. Enfin, nous concluons et présentons les perspectives de ce travail dans la section 5.

2 La RTO dédiée à la représentation de relations n-aires

Une RTO (McCrae et al., 2011; Reymonet et al., 2009; Roche et al., 2009) est une ressource comportant une composante conceptuelle (l'ontologie) et une composante terminologique, dans laquelle la manifestation linguistique (le terme) se distingue de la notion qu'elle dénote (le concept). D'après Reymonet et al. (2006), la modélisation d'une RTO est fortement influencée par la tâche, le domaine d'intérêt et l'application pour laquelle elle a été conçue. La tâche à réaliser est l'annotation et l'interrogation de relations n-aires ; le domaine d'intérêt est l'étude de données expérimentales quantitatives sachant que nous nous intéressons en

1. http ://www.w3.org/TR/swbp-n-aryRelations/ "Use Case 3 : N-ary relation with no distinguished participant"

particulier aux données dans le domaine des sciences du vivant ; l'application, enfin, est la construction d'un entrepôt de données ouvert sur le Web.

Nous présentons dans la suite la composante conceptuelle de la RTO dédiée à la représentation de relations n-aires, puis sa composante terminologique. Nous donnons ensuite une définition de notre RTO, et, enfin, une définition de sa cohérence structurelle.

2.1 La composante conceptuelle

La composante conceptuelle de la RTO est composée d'une ontologie noyau, appelée *core ontology*, qui permet de représenter des relations n-aires et d'une ontologie de domaine, appelée *domain ontology*, qui permet de représenter les concepts spécifiques à un domaine donné.

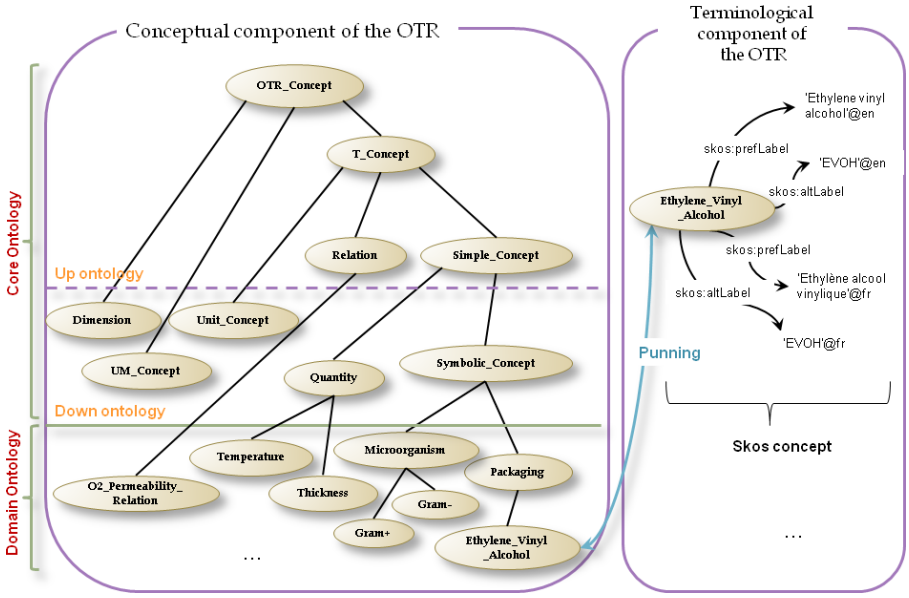

FIG. 2 – *Un extrait de* MapOpt OTR

Nous nous intéressons dans cet article aux relations n-aires entre des données quantitatives expérimentales. Nous avons ainsi découpé l'ontologie noyau en 2 sous-parties : une partie supérieure, appelée *Up ontology*, qui permet de représenter des relations n-aires entre n'importe quels arguments, et une partie inférieure, appelée *Down ontology*, qui permet de représenter des relations n-aires entre des données expérimentales quantitatives. La figure 2 donne un extrait d'une RTO définie dans le domaine du risque alimentaire microbiologique étendu aux emballages, appelée dans la suite *MapOpt OTR*. Pour définir l'ensemble des concepts de l'ontologie noyau (cf. définition 3), nous avons au préalable besoin d'introduire deux définitions :

Définition 1 *Pour un ensemble de concepts C, sa* hiérarchie de spécialisation*, notée H_C, est une relation acyclique, $H_C \subseteq C \times C$, telle que : si $(c_1, c_2) \in H_C$ alors c_1 est un sous-concept (un fils) de c_2, c_2 est un superconcept (ou père) de c_1, H_C^* est la fermeture réflexive, antisymétrique et transitive de H_C.*

Définition 2 *Etant donné un ensemble de concept C, H_C sa hiérarchie de spécialisation, et $ct \in C$, un concept donné, on définit l'*ensemble des sous-concepts stricts de *ct par :*
$C\mid_{ct} = \{c \in C \mid (c, ct) \in H_C^*\} \setminus \{ct\}$.

Définition 3 *L'ensemble des concepts de l'ontologie noyau de la RTO dédiée à la représentation de relations n-aires est définie par : $C_{core} = C_{core}^{Up} \cup C_{core}^{Down}$ où*
$C_{core}^{Up} = \{OTR\_Concept, T\_Concept, Relation, Simple\_Concept\}$, *et*
$C_{core}^{Down} = \{Symbolic\_Concept, Quantity, Unit\_Concept, Dimension\}$.

L'ontologie de domaine contient les concepts spécifiques à un domaine d'application particulier, ici le risque alimentaire microbiologique étendu aux emballages. Ils apparaissent dans la RTO comme des sous-concepts des concepts de l'ontologie noyau.

Définition 4 *L'ensemble des concepts de l'ontologie de domaine de la RTO dédiée à la représentation de relations n-aires est définie par : $C_{domain} = C\mid_{Relation} \cup C_{argument}$*
où $C_{argument}$=$C\mid_{Simple\_Concept} \setminus \{Quantity, Symbolic\_Concept\}$.

Remarque 1 *L'ensemble $C_{argument}$ contient les arguments d'une relation n-aire qui peuvent être des quantités ou des concepts symboliques pour une relation n-aire entre des données quantitatives expérimentales ou bien des concepts simples (i.e. des sous-concepts de* Simple_ Concept*) pour une relation n-aire entre n'importe quels arguments.*

En OWL, tous les concepts sont représentés par des classes OWL, qui sont organisées hiérarchiquement à l'aide de la relation de subsomption et sont deux à deux disjointes.

2.2 La composante terminologique

La composante terminologique de la RTO contient l'ensemble des termes du domaine étudié. Au moins un terme de cette composante est associé à chaque sous-concept du concept générique *T_Concept*. La RTO est représentée en OWL2-DL, ce qui permet via la technique du *punning* [2] de considérer chaque concept de la composante conceptuelle comme une instance du concept *Skos_Concept* et de lui associer sa terminologie grâce aux propriétés SKOS de labellisation (*skos :prefLabel*, *skos :altLabel*), recommandées par le W3C.

2.3 Définition de la RTO dédiée à la représentation de relations n-aires

Définition 5 *Une RTO dédiée à la représentation de relations n-aires est définie par le tuple suivant : $MRTO := (C, H_C, P, H_P, I, T, \mathcal{F})$ avec :*

- *C : l'ensemble des concepts de la partie conceptuelle de la RTO et H_C sa hiérarchie (cf. définition 1), où $C = C_{core} \cup C_{domain}$ (cf. définitions 3 et 4) ;*
- *P : l'ensemble des propriétés telles que définie par le W3C* [3]*, où une propriété est définie comme une relation binaire entre des instances de concepts, des littéraux et des types de données primitifs, et H_P sa hiérarchie telle que donnée dans la définition 1 où la notion de "concept" est remplacée par celle de "propriété" ;*

2. http ://www.w3.org/TR/owl2-new-features/#F12 :_Punning
3. http ://www.w3.org/TR/2004/REC-owl-guide-20040210/#SimpleProperties

– *T : l'ensemble des termes,* $T = \{t \mid t$ *est un littéral* $\}$ *;*
– *I : l'ensemble des instances de concepts ;*
– $\mathcal{F}$ *: un ensemble de fonctions retournant un sous-ensemble de termes, d'instances ou de concepts de la RTO.*

Nous présentons ci-dessous plusieurs fonctions de $\mathcal{F}$, utilisées dans la suite du papier :
– la fonction *signature* de $C\mid_{Relation}$ dans $2^{C_{argument}}$ qui permet d'associer à une relation $r \in C\mid_{Relation}$ le domaine de valeurs de chacun de ses arguments $(c_1, \ldots, c_n)$ où $c_i \in C_{argument}$;
– la fonction *dimension* de $C\mid_{Quantity}$ dans $I_{Dimension}$ [4] qui permet d'associer à une quantité $q \in C\mid_{Quantity}$ sa dimension $d \in I_{Dimension}$;
– la fonction *denote* de $C\mid_{T\_Concept}$ dans 2^T qui permet d'associer à un concept $c \in C\mid_{T\_Concept}$ son ensemble de termes $\{t_1, \ldots, t_n\} \in T$;
– la fonction *linkconcepts* de $C\mid_{T\_Concept} \times P \times$ {*allValuesFrom*, *someValuesFrom*, *hasValue*} dans $C\mid_{T\_Concept}$ qui permet d'associer à un concept $c_1 \in C\mid_{T\_Concept}$ un autre concept $c_2 \in C\mid_{T\_Concept}$ via la propriété $p \in P$ à l'aide d'une restriction de propriétés [5].

2.4 Définition d'une RTO structurellement cohérente

La RTO étant composée de concepts interdépendants (i.e. une relation et ses arguments), il est nécessaire d'identifier l'ensemble des contraintes qu'elle doit vérifier afin de préserver sa cohérence structurelle. En nous inspirant de Stojanovic (2004), nous introduisons la notion de RTO $\mathcal{CC}$-cohérente.

Définition 6 *Une RTO, définie par le tuple* $(C, H_C, P, H_P, I, T, \mathcal{F})$*, est dite* $\mathcal{CC}$-cohérente *si elle vérifie un ensemble de contraintes de cohérences structurelles, noté* $\mathcal{CC}$*.*

Nous avons identifié 39 contraintes de cohérences structurelles dont 16 portent sur les concepts, 7 sur les propriétés, 12 sur les instances et 4 sur les termes.

Nous présentons ci-dessous quelques exemples de ces contraintes :
– $\mathcal{CC}1$: une relation a au moins deux arguments : $\forall r \in C\mid_{Relation}, |signature(r)| \geq 2$
– $\mathcal{CC}2$: Tous les concepts appartenant à la signature d'une relation doivent être définis : $\forall r \in C\mid_{Relation}, \forall c \in signature(r) \implies c \in C_{argument}$
– $\mathcal{CC}3$: à chaque sous-concept de T_Concept est associé un terme dans la composante terminologique de la RTO : $\forall c \in C\mid_{T\_Concept}, \exists t \in T : t \in denote(c)$
– $\mathcal{CC}4$: Chaque quantité doit être associée à une seule valeur de dimension : $\forall q \in C\mid_{Quantity} \implies |dimension(q)| = 1$

3 Évolution de la RTO dédiée à la représentation de relations n-aires

Dans son cycle de vie, une ontologie peut être amenée à évoluer pour diverses raisons (Kondylakis et al., 2009) : (1) pouvoir prendre en considération de nouvelles données qui deviennent

4. Les dimensions sont définies comme des instances du concept *Dimension*.
5. http ://www.w3.org/TR/2004/REC-owl-guide-20040210/#PropertyRestrictions

pertinentes dans un domaine d'application ; (2) pouvoir tirer partie de nouvelles ressources ontologiques mises à disposition sur le Web concernant le domaine d'application ; (3) prendre en compte de nouveaux besoins des utilisateurs ; (4) améliorer la qualité des annotations qui dépend de la qualité de l'ontologie. Nous nous intéressons dans cet article à l'évolution de la structure de la RTO telle que donnée dans la définition 6, qui peut se traduire par divers changements tels que l'ajout d'une relation n-aire, l'ajout ou la suppression d'un de ses arguments, etc. Nous supposons dans la suite que les concepts de l'ontologie noyau de la RTO sont stables et que seule les concepts de l'ontologie de domaine et la terminologie peuvent évoluer.

Pour des raisons de simplicité et de lisibilité, nous présentons l'évolution de la structure de notre RTO à travers un cas d'utilisation, détaillé dans la sous-section 3.1, qui consiste à ajouter une nouvelle relation n-aire dans *MapOpt OTR*. Puis, nous présentons dans les sous-sections 3.2 et 3.3 les changements élémentaires et composés qui permettent une telle évolution.

3.1 Cas d'utilisation : ajouter une relation n-aire à *MapOpt OTR*

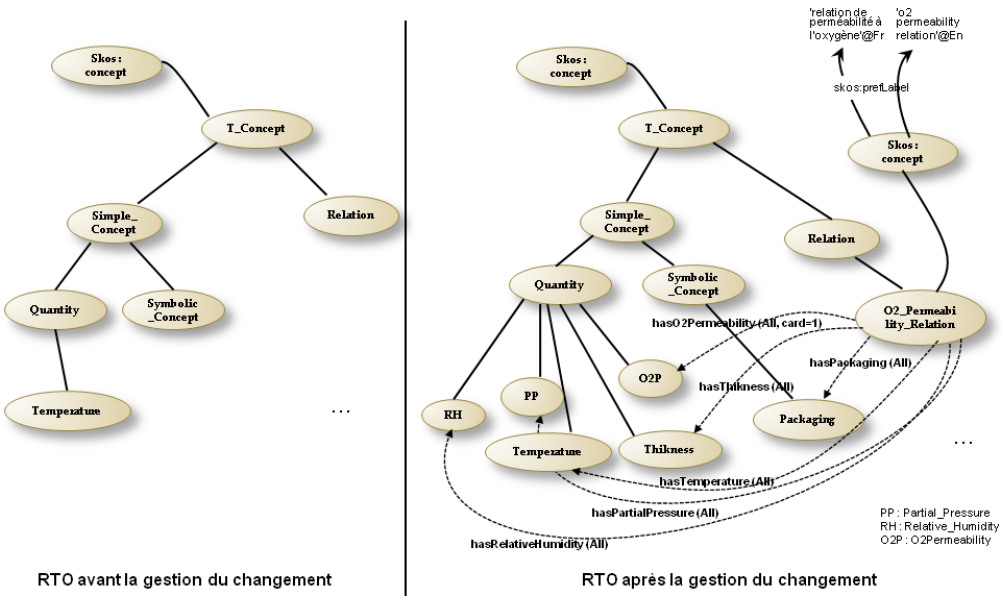

FIG. 3 – *Ajout de la relation O2Permeability_Relation dans MapOpt OTR.*

Supposons que la relation n-aire *O2Permeability_Relation* présentée dans la figure 1 ne soit pas définie dans *MapOpt OTR*. Pour pouvoir l'ajouter, le processus d'évolution doit permettre de transformer la structure de la RTO comme le montre la figure 3 : la partie gauche représente la RTO avant l'ajout de cette nouvelle relation et, celle de droite, la RTO après cet ajout. Comme la signature de la relation n-aire *O2Permeability_Relation* – (*Packaging*, *Temperature*, *Partial_Pressure*, *Relative_Humidity*, *Thickness*, *O2Permeability*) – contient six arguments, l'ontologue doit, avant de pouvoir ajouter la relation, commencer par ajouter le

nouveau concept *Packaging* comme sous-concept du concept *Symbolic_Concept* et les nouveaux concepts *Partial_Pressure*, *Relative_Humidity*, *Thickness* et *O2Permeability* comme sous-concepts du concept *Quantity*. L'ontologue doit ensuite définir le concept *O2Permeability_Relation* comme sous-concept du concept *Relation*, ainsi que les propriétés et les restrictions de propriétés permettant de relier la relation à ses arguments. Enfin, il doit définir la terminologie associée à chaque nouveau concept créé.

L'ontologue doit donc, pour ajouter une nouvelle relation n-aire dans la RTO, effectuer un nombre important d'opérations de modification. Il doit également veiller à ce que les modifications effectuées respectent les contraintes de cohérences présentées dans la sous-section 2.4. Afin d'alléger la tâche de l'ontologue et d'éviter des erreurs lors de manipulations fastidieuses de la RTO, il paraît nécessaire de définir un ensemble de changements élémentaires et composés permettant de faire évoluer la RTO tout en préservant sa $\mathcal{CC}$-cohérence.

3.2 Notion de changement élémentaire

Nous appelons changement élémentaire, une opération de modification de la RTO portant sur une seule entité (i.e. un concept, une propriété, une instance ou un terme dénotant un concept). Pour faire évoluer la RTO, nous avons identifié 60 changements élémentaires dont 8 s'appliquent aux concepts (e.g CreateHierarchyConceptLink), 41 aux propriétés (e.g CreateConceptsLinked) et 5 aux instances de concepts (e.g CreateInstance) et 6 aux termes (e.g. CreateTerm).

Comme l'a proposé Stojanovic (2004), la formalisation des changements repose sur la définition d'un ensemble de pré-conditions (un ensemble d'assertions qui doivent être vraies pour pouvoir appliquer le changement) et de post-conditions (qui correspondent au(x) résultat(s) du changement) pour chaque type de changement (i.e. élémentaire ou composé). Les pré-conditions associées à un changement correspondent au sous-ensemble de contraintes de cohérences $\mathcal{CC}$ portant sur les parties de la RTO affectées par le changement considéré. La vérification des pré-conditions d'un changement permet d'éviter d'appliquer ce changement sur une RTO non $\mathcal{CC}$-cohérente. Les post-conditions garantissent que le changement élémentaire a bien été appliqué, mais pas la $\mathcal{CC}$-cohérence de la RTO.

Nous présentons ci-dessous quatre changements élémentaires nécessaires pour faire évoluer la RTO *MapOpt OTR* dans le cas d'utilisation décrit dans la section 3.1.

Définition 7 ***CreateConcept***$(newc, c)$ *: :=*
Sémantique *: créer un nouveau concept* $newc$ *comme fils du concept* c
Pré-condition *:* $c \in C, newc \notin C_{domain}, (newc, c) \notin H_C$
Post-condition *:* $c \in C, newc \in C_{domain}, (newc, c) \in H_C$

Remarque 2 Rappelons que ce changement élémentaire ne peut s'appliquer qu'aux concepts de l'ontologie de domaine (i.e. $newc \in C_{domain}$), car l'ontologie noyau ne peut pas évoluer.

Exemple 1 *Pour faire évoluer MapOpt OTR (cf. figure 3), il faut appliquer six fois ce changement élémentaire, i.e. pour chaque nouveau concept, par exemple :*

```
CreateConcept(Packaging, Symbolic_Concept)
CreateConcept(Thickness, Quantity)
CreateConcept(O2_Permeability_Relation, Relation)
```

Définition 8 ***CreateProperty***$(newp, d, r)$ *: :=*
Sémantique *: créer une propriété* $newp$ *et lui associer un domaine* d *et un co-domaine* r
Pré-condition *:* $newp \notin P, d \in C, r \in C$
Post-condition *:* $newp \in P$*, domain(p)=d, co-domain(p)=r*

Exemple 2 *Pour faire évoluer MapOpt OTR, ce changement élémentaire doit être appliqué pour créer les propriétés reliant la relation n-aire à chacun de ses arguments, par exemple :*

```
CreateProperty(hasPackaging, Relation, Packaging)
CreateProperty(hasThickness, Relation, Thickness)
```

Définition 9 ***CreateTerm*** $(c, \{t_1, \ldots, t_n\})$ *: :=*
Sémantique *: associer les termes* $\{t_1, \ldots, t_n\}$ *au concept* c
Pré-condition *:* $c \in C \mid_{T\_Concept}, denote(c) = \emptyset$
Post-condition *:* $c \in C \mid_{T\_Concept}, denote(c) = \{t_1, \ldots, t_n\}$

Remarque 3 Si on veut pouvoir distinguer les labels préférés des labels alternatifs associés à un concept grâce aux propriétés SKOS de labellisation recommandées par le W3C, on peut dans la définition du changement élémentaire distinguer deux ensembles de termes comme suit : **CreateTermPrefAlt** $(c, \{pref_1, ..., pref_n\}, \{alt_1, ..., alt_m\})$.

Exemple 3 *Pour faire évoluer MapOpt OTR, ce changement élémentaire doit être appliqué pour créer la terminologie associée à chaque nouveau concept, par exemple :*

```
CreateTerm(Packaging, {Packaging@en, Emballage@fr})
CreateTerm(Thickness, {Thickness@en, Epaisseur@fr})
```

Définition 10 ***CreateLinkConcepts*** *(*c_1*, p,* c_2*, allValuesFrom) : :=*
Sémantique *: associer un concept* c_2 *au concept* c_1 *via une propriété p en utilisant la restriction allValuesFrom*
Pré-condition *:* $c_1, c_2 \in C_{domain}, p \in P, \{c \mid c \in C_{domain} \wedge linkconcepts(c_1, p, AllValues\text{-}From) = c\} = \emptyset$
Post-condition *:* $c_1, c_2 \in C_{domain}, p \in P, linkconcepts(c_1, p, allValuesFrom) = c_2$.

Ce changement est considéré comme élémentaire étant donné qu'il ne permet de manipuler qu'une seule entité : le concept c_1. Comme pré-condition, il s'agit de vérifier : i) que les concepts c_1 et c_2 et la propriété p sont bien définies et ii) qu'aucun argument n'est pas déjà associé directement au concept c_1 via la propriété p à l'aide de la restriction *allValuesFrom*.

Exemple 4 *Pour faire évoluer MapOpt OTR, ce changement doit être appliqué pour associer la relation* O2Permeability_Relation *à ses différents arguments, par exemple :*

```
CreateLinkConcepts(O2_Permeability_Relation, hasPackaging, Packaging,
allValuesFrom)
CreateLinkConcepts(O2_Permeability_Relation, hasThickness, Thickness,
allValuesFrom)
```

Il est important de noter que parmi les changements élémentaires que nous avons définis, certains ne sont pas accessibles à l'ontologue car ils produiraient une ontologie non $\mathcal{CC}$-cohérente. Par exemple, l'application du changement élémentaire *CreateConcept* par l'ontologue violerait la contrainte $\mathcal{CC}3$ car ce changement ne permet pas de créer la terminologie associée au nouveau concept. Ces changements élémentaires ne pourront donc être utilisés que par l'intermédiaire de changements composés.

3.3 Changements composés

La gestion de l'évolution d'une RTO demande généralement à l'ontologue de réaliser des opérations de changement de plus haut niveau que les changements élémentaires, appelés changements composés. Un changement composé est une opération de modification de la RTO qui affectent plusieurs entités, peut se décomposer en une succession de changements élémentaires et garantit que la RTO reste $\mathcal{CC}$-cohérente après son application. Pour faire évoluer la RTO, nous avons identifié 12 changements composés (e.g. AddConcept, AddHierarchyOfSymbolicConcept).

Nous présentons ci-dessous le changement composé nécessaire pour faire évoluer la RTO *MapOpt OTR* dans le cas d'utilisation décrit dans la section 3.1.

Définition 11 ***AddConcept***$(newc, c, d, \{u_1, \ldots, u_{n_u}\}, min, max, \{p_1, \ldots, p_n\}, \{o_1, \ldots, o_n\}, \{rs_1, \ldots, rs_n\}, \{t_1, \ldots, t_m\})$
Pré-condition *:* $newc \notin C_{domain}, (newc, c) \notin H_C, d \in I_{Dimension}$, $[(c \in C \mid (c, Relation) \in H_C^*) \wedge (|\{o_1, \ldots, o_n\}| \geq 2)] \bigvee (c \in C \mid (c, Simple\_Concept) \in H_C^*)$, $u_1, \ldots, u_{n_u} \in I_{Unit\_Concept}$ [6], $p_1, \ldots, p_n \in P$, $o_1, \ldots, o_n \in C_{argument}$, $rs_1, \ldots, rs_n \in$ *{allValuesFrom, someValuesFrom, hasValue}*
Sémantique *: ajouter un concept* $newc$*, qui peut être une relation (son père* c *est tel que* $c \in C \mid (c, Relation) \in H_C^*$*) ou un concept simple (son père* c *est tel que* $c \in C \mid (c, Simple\_Concept) \in H_C^*$*) qui sera utilisé comme argument d'une relation, avec sa hiérarchie, sa terminologie et ses restrictions*
Traitement du concept *:*

- *Créer le concept* $newc$ *:* $CreateConcept(newc, c)$
- *Créer la terminologie associée à* $newc$ *:* $CreateTerm(newc, \{t_1, \ldots, t_m\})$

Traitement des restrictions :

1. ***CreateLinkDimension*** $(newc, hasDimension, d, hasValue)$
2. $\forall i \in [1, n_u]$, ***CreateLinkUnit*** $(newc, hasUnitConcept, u_i, allValuesFrom)$
3. ***CreateRestrictionMinMax*** $(newc, hasNumericalValue, min, max)$
4. $\forall i \in [1, n]$, *(Si* rs_i *=* $allValuesFrom$ *alors*
   CreateLinkConcepts $(newc, p_i, o_i, allValuesFrom)$
   Sinon si rs_i *=* $someValuesFrom$
   CreateLinkConcepts $(newc, p_i, o_i, someValuesFrom)$
   Sinon Si rs_i *=* $hasValue$
   CreateLinkConcepts $(newc, p_i, o_i, hasValue)$*).*

Post-condition *:* $newc \in C_{domain}, (newc, c) \in H_C$

6. Les unités de mesure sont définies comme des instances du concept *Unit_Concept*.

Remarque 4 Les pré-conditions des changements élémentaires permettent de tester l'existence de leurs arguments. Par exemple, le changement élémentaire **CreateLinkConcepts** ne s'applique dans le traitement des restrictions 4 que si $newc \in C_{domain}, p_i \in P, o_i \in C_{domain}$ (cf. définition 10).

Le changement composé **AddConcept** permet de créer soit une nouvelle relation, soit un nouveau concept, sous-concept de *Simple_Concept*, (i.e. une quantité, un concept symbolique ou un concept simple), qui sera utilisé comme argument d'une relation. Pour tous ces concepts, il y a un traitement en commun : créer le nouveau concept, le lier avec son concept père et créer sa terminologie. Ensuite, il n'y a aucun traitement supplémentaire pour les concepts simples et les concepts symboliques. Pour les quantités, il s'agit de leurs associer leur dimension (traitement des restrictions 1), leurs unités de mesures (traitement des restrictions 2) et éventuellement un intervalle de valeurs (traitement des restrictions 3). Enfin pour les relations, il s'agit de créer les liens avec leurs différents arguments ($\{o_1, \ldots, o_n\}$) selon les types de restriction ($\{rs_1, \ldots, rs_n\}$) demandés (traitement des restrictions 4). Lors de l'ajout d'un nouveau concept à l'aide du changement composé **AddConcept**, le système peut déterminer le type du concept à ajouter (relation ou concept simple) en fonction de sa liste de paramètres.

On remarquera que ce changement composé permet de créer non seulement des relations n-aires entre des données quantitatives expérimentales, mais également entre n'importe quels types d'arguments, que nous avons identifiés comme étant des concepts simples.

Exemple 5 *Finalement, les changements à effectuer pour faire évoluer* MapOpt OTR *en y ajoutant la relation* O2Permeability_Relation *comme décrit dans la section 3.1 sont :*
i) Ajouter le concept symbolique Packaging *:*

```
AddConcept(Packaging, Symbolic_Concept, , {}, , , {}, {}, {}, {Packaging@en,
emballage@fr});
```

ii) Ajouter la quantité Partial_Pressure *:*

```
AddConcept(Partial_Pressure, Quantity, pressure or stress-dimension, {Atmosphere,
Hectopascal, Millibar, Pascal, Percent}, , , {}, {}, {}, {Partial pressure@en, pression
partielle@fr})
```

iii) Ajouter la quantité Relative_Humidity *:*

```
AddConcept (Relative_Humidity, Quantity, dimension-one, {One, Percent}, 0,100, {}, {}, {},
{Partial pressure@en,
pression partielle@fr})
```

iv) Ajouter la quantité Thickness *:*

```
AddConcept (Thickness, Quantity, length-dimension, {Centimeter, Decimeter, Femtometer,
Meter, Micrometer, Millimeter, Nanometer, Picometer, Yotometer, Zeptometer}, 0, , {}, {},
{}, {Thickness@en, Epaisseur@fr})
```

v) Ajouter la quantité O2Permeability *:*

```
AddConcept (O2Permeability, Quantity, dimension-one, {Barrer, Mole_Per_Meter_Per_Second_
Per_Pascal, Milli- liter _Per_100_Square_Inch_Per_Day}}, , , {}, {}, {}, {O2
permeability@en, Perméabilité à l'oxygène@fr})
```

vi) Ajouter les propriétés :

```
CreateProperty(hasPackaging, Relation, Packaging)
CreateProperty(hasTemperature, Relation, Temperature)
CreateProperty(hasPartialPressure, Relation, PartialPressure)
CreateProperty(hasRelativeHumidity, Relation, RelativeHumidity)
CreateProperty(hasThickness, Relation, Thickness)
CreateProperty(hasO2Permeability, Relation, O2Permeability)
```

vii) Ajouter enfin la nouvelle relation O2Permeability_Relation *:*

```
AddConcept (O2Permeability_Relation, Relation, , {}, , , {hasPackaging, hasTemperature,
hasPartielPressure, hasRelativeHumidity, hasThickness, hasO2Permeability}, {Packaging,
Temperature, PartielPressure, RelativeHumidity, Thickness, O2Permeability}, {allValuesFrom,
allValuesFrom, allValuesFrom, allValuesFrom, allValuesFrom, allValuesFrom}, {O2
permeability relation@en, relation de perméabilité à l'oxygène@fr});
```

4 Comparaison avec l'état de l'art

Plusieurs approches d'évolution d'ontologie ont été proposées dans la littérature. Le travail le plus significatif est présenté dans Stojanovic (2004). L'auteur a proposé un processus global de gestion d'évolution pour les ontologies : KAON. Ce processus assure la cohérence de l'ontologie ainsi que de ses artefacts dépendants après application des changements ontologiques. Les méthodes de résolution d'incohérences se basent sur les contraintes du langage KAON.
Une deuxième approche, Noy et al. (2004), permet de gérer les versions d'une ontologie. Le versionnement permet de conserver et d'accéder aux différentes versions de l'ontologie générées au cours de l'évolution de celle-ci. Il s'agit de déterminer les relations sémantiques existantes entre les entités ontologiques de deux versions afin de pouvoir détecter les changements produites lors de l'évolution. Cette méthodologie se limite à la gestion des versions d'une ontologie après son évolution sans supporter le processus d'évolution lui même.
Le travail de Luong (2007) s'intéresse plus spécifiquement à l'impact de l'évolution d'ontologies sur les annotations sémantiques. Pour détecter les annotations incohérentes l'auteur propose deux méthodes. La première ne peut s'appliquer que si une trace des évolutions de l'ontologie a été conservée et la deuxième s'applique en l'absence de cette trace.
Dans (Tissaoui et al., 2011), les auteurs proposent une approche de gestion des évolutions d'une ontologie en prenant en compte sa composante lexicale. Cette approche consiste, d'une part, à gérer les annotations de documents qui sont devenues incohérentes suite à l'évolution de la RTO, et d'autre part, à faire évoluer la RTO suite à une annotation de nouveaux documents.
Les différentes approches présentées dans la littérature permettent de gérer l'évolution d'une ontologie ou d'une RTO mais aucune d'entre elles ne traite l'évolution des relations n-aires qui peuvent être représentées dans une ontologie. La gestion de l'évolution des relations n-aires nécessite donc un traitement particulier.

5 Conclusion

Nous avons présenté dans cet article les définitions des changements élémentaires et composés permettant de faire évoluer la structure d'une RTO dédiée à la représentation de relations n-aires tout en maintenant sa cohérence structurelle. Nous nous sommes en particulier intéressés aux relations n-aires entre des données quantitatives expérimentales. Pour des questions de place, nous avons focalisés notre présentation sur les changements nécessaires pour ajouter une nouvelle relation n-aire à notre RTO.

Dans un avenir proche, nous continuerons d'étudier l'évolution d'une RTO dédiée à la représentation de relations n-aires en proposant des stratégies d'évolution associées aux changements. Ces stratégies permettent de prendre en considération les différents choix que peut faire un ontologue. Par exemple, pour gérer les sous-concepts d'un concept c supprimé, l'ontologue peut choisir soit de les supprimer, soit de les associer au père, soit de les associer au

concept racine. La prise en compte de ces choix nécessite un traitement à part. Nous étudierons, aussi, l'évolution des unités de mesures qui sont liées aux quantités. La difficulté dans la gestion de l'évolution des unités de mesure s'explique par le fait qu'elles sont elles-même décomposables, ce qui va nous amener à devoir traiter des petits blocs d'ontologies qui sont très fortement interconnectés.

Références

Kondylakis, H., G. Flouris, et D. Plexousakis (2009). Ontology and Schema Evolution in Data Integration : Review and Assessment. In *OTM Conferences (2)*, pp. 932–947.

Luong, P. H. (2007). *Gestion de l'évolution d'un Web sémantique d'entreprise*. These, École Nationale Supérieure des Mines de Paris.

McCrae, J., D. Spohr, et P. Cimiano (2011). Linking Lexical Resources and Ontologies on the Semantic Web with Lemon. In *ESWC (1)*, Volume 6643 of *LNCS*, pp. 245–259. Springer.

Noy, N. et A. Rector (2006). Defining N-ary Relations on the Semantic Web.

Noy, N. F., S. Kunnatur, M. C. A. Klein, et M. A. Musen (2004). Tracking changes during ontology evolution. In S. A. McIlraith, D. Plexousakis, et F. van Harmelen (Eds.), *International Semantic Web Conference*, Volume 3298 of *Lecture Notes in Computer Science*, pp. 259–273. Springer.

Reymonet, A., N. Aussenac-Gilles, et J. Thomas (2006). Tâche, domaine et application : influences sur le processus de modélisation de connaissances. In *Actes d'IC*, pp. 71–80.

Reymonet, A., J. Thomas, et N. Aussenac-Gilles (2009). Ontology based information retrieval : an application to automotive diagnosis. In *International Workshop on Principles of Diagnosis*, pp. 9–14.

Roche, C., M. Calberg-Challot, L. Damas, et P. Rouard (2009). Ontoterminology - a new paradigm for terminology. In *KEOD*, pp. 321–326.

Stojanovic, L. (2004). *Methods and Tools for Ontology Evolution*. Ph. D. thesis, University of Karlsruhe, Germany.

Summary

In this paper we are addressing the problem of the evolution of an Ontological and Terminological Resource (OTR) allowing to represent n-ary relations. During its life cycle, an OTR may need to change for several reasons (e.g. new available information, additional users' needs, mapping to new available ontologies). In this paper, we present the formal definitions of changes to be applied to our OTR, modifying its structure while maintaining its structural consistence. Our examples are using an OTR representing n-ary relations between quantitative experimental data in the field of microbiological risk in food, with its surrounding packaging.