

G-OWL : Vers un langage de modélisation graphique, polymorphique et typé pour la construction d'une ontologie dans la notation OWL

Michel Héon, Roger Nkambou

► To cite this version:

Michel Héon, Roger Nkambou. G-OWL : Vers un langage de modélisation graphique, polymorphique et typé pour la construction d'une ontologie dans la notation OWL. IC - 24èmes Journées francophones d'Ingénierie des Connaissances, Jul 2013, Lille, France. 2013. <hal-01104001>

HAL Id: hal-01104001

<https://hal.inria.fr/hal-01104001>

Submitted on 15 Jan 2015

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.



Distributed under a Creative Commons Attribution| 4.0 International License

G-OWL : Vers un langage de modélisation graphique, polymorphique et typé pour la construction d'une ontologie dans la notation OWL

Michel Héon¹, Roger Nkambou²

¹Université du Québec à Montréal, Montréal, Canada
heon@cotechnoe.com

²Université du Québec à Montréal, Montréal, Canada
nkambou.roger@ugam.ca

Résumé : Le *Web Ontology Language* (OWL) standardisé par le W3C a pour objectif d'offrir un langage de conception d'ontologies pour le web sémantique. L'ingénierie d'une ontologie est une activité complexe nécessitant une habilité peu accessible à des experts de contenu. En revanche, pour modéliser du contenu métier, la modélisation graphique semi-formelle est une technique souvent employée pour offrir un outil de représentation des connaissances à des experts de contenu peu familier au processus de conception d'une ontologie. Dans cet article, nous présentons de quelle manière l'usage du polymorphisme et le typage des symboles du vocabulaire graphique permettront de concevoir le langage G-OWL, un langage graphique qui vise à permettre la représentation de connaissances métiers dans le formalisme OWL pour des non-experts de l'ingénierie ontologique.

Mots-clés : OWL, ontologie, modèle graphique, langage ontologique graphique, langage graphique, ontologie graphique, cartographie de connaissances, web sémantique.

1 Introduction

La construction d'une ontologie de type OWL est une activité formelle difficilement accessible à un expert de contenu. En revanche, la modélisation graphique semi-formelle est une solution souvent envisagée pour permettre à des experts de contenu de représenter graphiquement leur connaissance métier. Pensons par exemple à l'utilisation des langages tels que : le *Mind Mapping* (Buzan & Buzan, 1994), le *Concept Mapping* (Novak & Cañas, 2006) ou encore le langage de *Modélisation par Objets Typés* (MOT) (Paquette, 2010) qui ont su démontrer leur efficacité dans la représentation de connaissances. Le langage graphique semi-formel est parfois employé pendant l'étape de conception d'un système (Rumbaugh *et al.*, 1999) ou encore pour favoriser le transfert de connaissances dans les organisations (Basque & Pudelko, 2010) ou

l'apprentissage dans des situations éducatives (Kinshuk *et al.*, 2008; Paquette, 2002). Il est aussi employé pour susciter les échanges pendant une séance de remue-méninges (Buzan & Buzan, 1994) ou plus simplement pour représenter graphiquement un énoncé. Certaines études (Basque & Pudenko, 2010) tendent à démontrer que l'usage de langage semi-formel facilite l'explicitation de connaissances tacites puisque la spontanéité n'est pas bloquée par une charge cognitive trop lourde associée à une formalisation de la pensée. Le désavantage de l'utilisation d'un langage semi-formel est que la carte de connaissances produite avec ce langage ne peut pas être directement exploitée par un système automatique de traitement de la connaissance sans que celle-ci soit préalablement désambiguïsée.

Deux besoins sont ainsi confrontés. D'une part, le besoin de l'expert de contenu qui aspire à l'utilisation d'un langage de représentation de connaissances qui soit expressif, simple d'usage et qui stimule sa créativité, et d'autre part, le besoin du cognitifien, d'utiliser un langage de représentation de la connaissance de degré formel pour la conception de systèmes logiciels intelligents ou d'une mémoire formelle d'entreprise.

Dans cet article, nous proposons un langage qui vise à réduire l'apparent espace qui sépare le besoin de l'expert de contenu de celui du cognitifien. Le but de G-OWL (acronyme de *Graphic OWL*) est de fournir un langage, qui est : graphique, simple d'usage (en comparaison avec l'usage direct du OWL) et qui permet de produire une ontologie formelle dans le formalisme de l'OWL.

1.1 Historique

Actuellement, quelques langages permettent l'édition graphique d'ontologies formelles. L'*Ontology Definition Metamodel* de Gašević *et al.* (2006) et son implantation informatique l'*EMF Ontology Definition Metamodel (EODM)* (Yang, 2006) sont des métamodèles de représentation d'OWL dans le formalisme UML. Les éléments syntaxiques d'OWL sont cartographiés en UML par l'attribution de « *stéréotypes* » aux classes et aux relations utilisées. Le modèle UML est ensuite converti en OWL.

De même, dans le domaine des graphes conceptuels (GC), les travaux de Sowa (1999) ont permis la mise au point d'un langage formel et graphique de représentation d'ontologies. Le GC appuie la représentation graphique d'ontologie sur la logique des prédicats du premier ordre qui va bien au-delà de la logique des descriptions (Baader *et al.*, 2007) sur laquelle se fonde l'OWL. À l'instar du graphe *entité-relation* de Chein (2008), une notation en graphe du OWL couramment utilisée consiste à représenter la classe et l'individu en entité du graphe et représenter le prédicat en relation du graphe (voir l'exemple de la figure 1a, section 4).

En fait, de notre point de vue, cette représentation en graphe comporte quelques manquements importants : d'abord, elle ne met pas en évidence la distinction marquée entre : des entités et des relations qui sont propres au langage ontologique, des entités et des relations associés à la sémantique du domaine (par ex. : `rdf:type` *versus* `vin:hasColor`). Ensuite, cette notation est mal adaptée à la représentation signifiante des restrictions ou des axiomes.

Bien que l'ODM, le GC et le graphe entité-relation soient des langages de représentation d'une ontologie de type graphique, ils ne sont pas des langages simples d'usage pour un non-informaticien. À vrai dire, leur usage fondé sur la représentation formelle du domaine du discours impose, à l'utilisateur, la connaissance approfondie de l'ingénierie logicielle et de la modélisation formelle.

D'un autre côté, il existe des méthodes et des outils qui visent à construire une ontologie reposant sur la représentation graphique et semi-formelle de la connaissance. Par exemple, Eskridge (2006) propose une technique de formalisation OWL à partir d'une représentation en *Concept mapping*. Plus récemment OntoCASE (Héon, 2010, 2012) présente une méthodologie assistée par un outil logiciel qui formalise un modèle MOT semi-formel en une ontologie OWL. Il contient un éditeur de modèle MOT (eLi), un système expert à la désambiguïsation ainsi qu'un système expert à la conversion du modèle MOT en ontologie. Le point à souligner de ces méthodes est qu'à l'étape de formalisation du modèle semi-formel, l'humain doit intervenir pour compléter le processus de désambiguïsation du symbole semi-formel. Le processus de formalisation ne peut donc pas être complètement automatisé.

1.2 Hypothèse

Une caractéristique importante qui assure la convivialité d'un langage semi-formel est le nombre restreint de symboles combiné à une simplification des règles grammaticales. En contrepartie, ces simplifications engendrent un coût au niveau de l'expressivité du langage. Ce coût est compensé par la polysémie (surcharge sémantique) des symboles qui permet d'accroître l'expressivité du langage. Lorsque la polysémie est trop forte, la désambiguïsation du langage se complexifie. Après un certain point, la désambiguïsation *ne peut plus être automatisée*. C'est à partir de ce point que le langage porte la dénomination de *semi-formelle*.

Deux techniques permettront d'accroître la polysémie de G-OWL afin de maintenir au minimum le nombre de symboles tout en conservant une expressivité identique à celle du OWL, et sans pour cela, que G-OWL devienne un langage semi-formelle:

- 1) le **polymorphisme** permettra de donner plusieurs significations à un symbole graphique. Le symbole pourra ultérieurement être désambiguïsé par leur agencement topologique. Par exemple : si un *LienS* est entre deux *concepts*, alors il sera désambiguïsé en *subClassOf*. Tandis que si le *LienS* est entre deux *attributs* il sera désambiguïsé en *subPropertyOf*.
- 2) l'assignation **typologique** est la technique qui limite le nombre de symboles en lui assignant un type. Cette technique, employée pour la conception du langage MOT attribut un type aux entités et relations du langage. De plus, à chaque type correspond un nombre limité de significations. Pendant la modélisation, ce typage offre l'avantage de clairement distinguer la sémantique associée au langage de la sémantique associée au domaine représenté dans le modèle. Lors de la formalisation, le type sert de guide au processus de désambiguïsation.

2 Spécification de G-OWL

La spécification de G-OWL définit le vocabulaire graphique du langage G-OWL. Elle identifie les entités typées ainsi que les relations typées qui constituent l'alphabet du vocabulaire. La deuxième partie de la spécification permet de définir, en fonction de la polysémie, le polymorphisme et le typage ontologique du vocabulaire de G-OWL. Pour chaque élément et relation du vocabulaire, on associe un nombre fini de constructeurs OWL possibles.

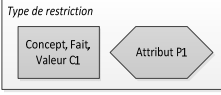
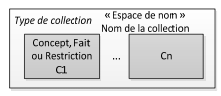
2.1 Vocabulaire graphique de G-OWL

Présenté au tableau 1 ainsi qu'au tableau 2, l'alphabet de G-OWL se compose d'entités typées graphiques ainsi que de relations typées qui permettent de relier les entités d'un modèle.

Les entités sont regroupées en deux niveaux d'abstraction, le niveau *abstrait* et le niveau *factuel*. Le niveau abstrait renvoie à l'idée de la *Terminological Box* (TBox) en logique de description (DL) (Baader, *et al.*, 2007). La TBox est l'espace de modélisation qui permet de représenter le niveau terminologique d'un sujet. On pourrait aussi dire qu'il fait référence aux abstractions d'un domaine du discours. Par exemple, pour le domaine de la *Famille*, c'est dans le niveau abstrait que se définissent les concepts tels que : *Père*, *Mère*, *Enfant*, *Parent*, et les propriétés telles que : *estEnfantDe*, *etc.* Quant au niveau factuel, qui renvoie en DL à l'idée de l'*Assertional Box* (ABox), il permet de représenter les *faits* et les *prédicats* qui unissent les faits concernant un domaine du discours. Le fait est une entité observable de la réalité. Ainsi, toujours dans l'exemple du domaine de la *Famille*, l'alphabet du niveau factuel est utilisé pour représenter des faits tels que *MARIE*, *MARC*, *etc.*

Le niveau factuel permet aussi de représenter la mise en relation de deux faits par la déclaration d'un énoncé selon la structure *Sujet/Prédicat/Objet* (par exemple : *MARC estEnfantDe MARIE*).

Tableau 1 : Alphabet des entités de G-OWL

	Type d'entité	Alphabet graphique	Signification
Niveau abstrait	gowl:Concept	« Espace de nom » Nom du concept	Le rectangle est utilisé pour représenter un <i>Concept</i> qui désigne le « quoi » des choses
	gowl:Restriction	Type de restriction 	Le rectangle conteneur est utilisé pour représenter une <i>Restriction</i> universelle, existentielle ou de valeur. Il est aussi utilisé pour représenter la cardinalité associée à une propriété.
	gowl:Collection	Type de collection 	Le rectangle conteneur est aussi utilisé pour représenter une <i>collection</i> de connaissances déclaratives.
	gowl:Attribut	(Type d'attribut) « Espace de nom » Nom de l'attribut	L'hexagone est utilisé pour représenter un attribut qui définit la propriété entre des concepts.
Niveau factuel	gowl:Fait	« Espace de nom » Nom du fait	Le rectangle pointillé est utilisé pour représenter un fait, un objet observable de la réalité, un individu OWL
	gowl:Donnee	(Type de donnée) Valeur	Le rectangle pointillé est aussi utilisé pour représenter une donnée de type <i>entier, réel, caractères</i> , etc.

Pour chaque élément graphique qui composent l'alphabet de G-OWL, on peut attribuer un nom qui sert d'identifiant unique. Dans la plupart des cas, il est permis d'attribuer un *espace de nommage* pour identifier au besoin le domaine d'appartenance du sujet représenté. La *restriction*, la *collection* ainsi que l'*attribut* sont des éléments du vocabulaire qui sont *typés*. Les valeurs possibles de chaque type sont présentées au tableau 4.

Le vocabulaire du niveau abstrait comprend un alphabet de quatre entités graphiques:

- Le gowl:Concept représente le « quoi » des choses, il sert à décrire l'essence d'un objet concret. Le concept est identifié par le *nom du concept* et optionnellement par l'indication de l'*espace de nommage* auquel il appartient.
- Le gowl:Restriction est utilisé pour caractériser les individus d'une classe. La restriction est un concept anonyme qui met en relation un attribut un concept et un fait selon le cas. Le tableau 4 présente les *types de restrictions* possibles.
- Le gowl:Collection est utilisé pour définir un regroupement de concepts. Le regroupement peut être conditionné par un

opérateur logique tel que l'union ou l'intersection. Les *types de collection* sont définis au tableau 4.

- Le `gowl:Attribut` désigne les propriétés qui existent entre des concepts, des restrictions ou des relations. L'identification de l'attribut est réalisée par le *nom de l'attribut* et optionnellement par l'identification de l'*espace de nom*. Il existe plusieurs *types d'attributs* qui sont présentés par les symboles du tableau 4.

Le vocabulaire du niveau factuel comprend quant à lui deux entités graphiques :

- Le `gowl:Fait` représente un élément de la réalité observable. Il est l'instance d'un concept. Le fait est identifié par le *nom du fait* et il peut aussi appartenir à un domaine qui est identifié par l'*espace de nom*.
- Le `gowl:Donnee` sert à représenter une donnée de type Entier, Réelle ou une Chaîne de caractères. Le type de donnée est indiqué entre parenthèses '()'. Le libellé du rectangle représente la *valeur* de la donnée.

Chaque élément alphabétique du langage G-OWL peut être relié à un autre élément alphabétique par une relation typée prédéfinie ou dynamiquement définie par l'association du nom du lien à un attribut déjà existant. Le tableau 2 présente les différentes relations typées ainsi que leur signification alors que le tableau 3 présente la synthèse des règles d'utilisation des liens typés de G-OWL.

Tableau 2 : Vocabulaire des relations de G-OWL

Type de relation	Signification
<code>gowl:LienS</code> --- S -->	Le lien de <i>Spécialisation</i> associe deux connaissances de niveau abstrait de même type dont le premier est une spécialisation de la seconde.
<code>gowl:LienDS</code> (LienS doublement orienté) <-- S -->	Le lien de <i>synonymie</i> associe deux connaissances de niveau abstrait de même type ou deux faits. Elle indique que la première connaissance est l'équivalent (ou le synonyme) de la seconde.
<code>gowl:LienA</code> --- A -->	Le lien d'attribution associe un attribut à un concept, une restriction ou une collection afin de préciser l'image ou le domaine d'une propriété.
<code>gowl:LienNT</code> --- nom d'un Attribut -->	Le lien <i>NonTypé</i> est utilisé pour associer un <i>prédicat</i> entre un fait et une connaissance de niveau factuel. Le nom du prédicat est associé à un attribut existant par le <i>nomDunAttribut</i> .
<code>gowl:LienI</code> --- I -->	Le lien d' <i>instanciation</i> associe un concept à un fait qui caractérise une instance de cette connaissance.
<code>gowl:LienD</code> ---D-->	Le lien de <i>Différence</i> est utilisé pour indiquer la disjonction entre deux connaissances déclaratives abstraites. Il est aussi utilisé pour différencier deux faits.
<code>gowl:LienIO</code> ---`l-->	Le lien <i>Inverse</i> associe un attribut inverse à un attribut déjà existant.

L'usage des relations typées est conditionné par des règles d'utilisation de la relation. Par exemple, l'emploi du LienS (*sorte de*) est conditionné par une règle qui indique que la connaissance à la *source* et à la *destination* du LienS doivent être toutes les deux du même type. En se référant au tableau 3 on peut déduire qu'un concept peut être lié par un LienS à un autre concept, à une restriction ou une collection.

Tableau 3 : Synthèse des règles d'utilisation des relations typées de G-OWL

Destination	Concept	Attribut	Fait	Restriction	Collection
Source					
Concept	S, DS, I, D	A	I	S, DS, D	S, DS, D
Attribut	A	S, DS, -1		A	A
Fait			NT		
Restriction	S, DS	A	I		
Collection	S, DS	A	I		

2.2 Polysémie et polymorphisme ontologique de G-OWL

Grâce à la polysémie et au polymorphisme, G-OWL restreint son alphabet à l'usage de cinq entités et de sept relations conditionnées par des règles d'usage explicitement définies en comparaison avec le langage OWL qui étend son usage à une trentaine de constructeurs dont la manipulation nécessite l'usage de règles implicites. Le tableau 4 présente la polysémie des éléments du vocabulaire de G-OWL.

Pour chaque élément de l'alphabet de G-OWL, il existe un ensemble restreint et fini de symboles qui permet de *typer* l'élément afin de le désambiguïser. La désambiguïstation peut aussi être réalisée de manière automatique par l'identification de la nature des éléments qui sont mis en relation (désambiguïstation topologique). Figuré à la fig.1, un synonyme (le `gowl:LienDS`) sera désambiguïsé par `owl:sameAs` si le lien relie deux faits (`gowl:Fait`) alors qu'il sera désambiguïsé par `owl:equivalentClass` s'il relie deux concepts (`gowl:Concept`) ou un concept à une restriction.

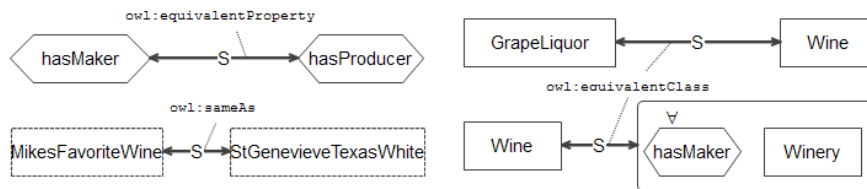


Fig. 1 –Polymorphisme du LienDS

Tableau 4 : Polysémie de l'alphabet de G-OWL

Alphabet polymorphe de G-OWL	Polysémie dans la notation OWL	Désambiguïsation de type et topologique
gowl:Concept	owl:Class	
gowl:Attribut	owl:DatatypeProperty owl:ObjectProperty owl:TransitivProperty owl:SymetricProperty owl:FunctionalProperty owl:InverseFunctional	si le codomaine est une donnée si le codomaine est un fait T S F I
gowl:Fait	xsd:Boolean xsd:string xsd:integer xsd:float <i>individu</i>	Bool String Int Float
gowl:Restriction	owl:Restriction owl:someValuesFrom owl:allValuesFrom owl:hasValue owl:minCardinality owl:maxCardinality owl:cardinality owl:onProperty	\exists \forall \ni \leq \geq $=$
gowl:Collection	owl:intersectionOf owl:unionOf owl:oneOf owl:DataRange owl:complementOf owl:distinctMembers owl:AllDifferent	\cap \cup [] [] \neg \neq
gowl:LienS	rdfs:subClassOf rdfs:subPropertyOf	si le lien est entre deux concepts si le lien est entre deux propriétés
gowl:LienDS	owl:equivalentClass owl:equivalentProperty owl:sameAs	Si le lien est entre deux attributs Si le lien est entre deux attributs Si le lien est entre deux faits
gowl:LienA	rdfs:domain rdfs:range	Si la source du lien est concept et que sa destination est un attribut Si la source du lien est attribut et que sa destination est un concept
gowl:LienNT	<i>Prédicat</i>	Une chaîne de caractère
gowl:LienI	rdfs:typeOf	I
gowl:LienD	owl:disjointWith owl:differentFrom	Si le lien est entre deux concepts Si le lien est entre deux faits
gowl:LienIO	owl:inverseOf	-1

3 Test d'expressivité de G-OWL

Le but de ce test est de démontrer qu'il est possible de décrire l'ensemble des éléments d'expressivité soutenu par l'OWL. Pour ce faire, la section est structurée selon la classification des constructeurs tels que présentés par McGuinness (2004) et Yu (2011) en utilisant l'ontologie *Wine.owl* (Deborah L. McGuinness *et al.*, 2004) en tant qu'exemple présenté par l'auteur afin d'illustrer les divers aspects de l'expressivité d'OWL.

Dans l'ontologie sont emmagasinés les éléments qui servent à représenter le domaine du discours. On y retrouve la classe, la propriété, la restriction sur les propriétés telle que la restriction universelle et existentielle, la restriction de cardinalité ou encore la restriction de valeur; la relation de synonymie; la collection sur les opérateurs booléens tels que l'intersection ou l'union de classes; la disjonction et l'énumération d'objets. Finalement, la représentation des faits du domaine du discours.

3.1 Éléments de base en G-OWL : la classe et la propriété

Le tableau 5¹ présente en G-OWL ainsi qu'en OWL N-3 la définition de *CabernetSauvignonGrape*. Selon l'ontologie, le *CabernetSauvignonGrape* est un *WineGrape* qui lui-même est une sorte de *Grape*. Le `gowl:Concept Grape` est défini dans l'espace de nommage `food`. Lors de la traduction du modèle G-OWL en ontologie OWL, les `gowl:Concept WineGrape` et *Grape* seront traduits en classe nommée (`owl:Class`); le `gowl:Fait CabernetSauvignonGrape` sera traduit en ressource `rdf`; le `gowl:LienS` sera traduit par la propriété `owl:subClassOf` et le `gowl:LienI` par `rdf:type`.

Le `gowl:LienS`, qui uni ces propriétés, est traduit en `rdfs:subPropertyOf` par la règle de désambiguïsation topologique qui stipule qu'un `gowl:LienS` entre deux `gowl:Attribut` sera traduit en `rdfs:subPropertyOf`. Le `gowl:LienA` est aussi utilisé de manière polysémique pour unir un `gowl:Attribut` à un `gowl:Concept` et *vice-versa*. La polysémie de ce lien est levée par l'application de la règle de désambiguïsation topologique qui stipule qu'un `gowl:LienA` ayant à sa source un `gowl:Concept`, et à sa destination un `gowl:Attribut` sera traduit par `rdfs:range` alors qu'il sera traduit par `rdfs:domain` dans le cas où la source et la destination du `gowl:LienA` sont inversées.

¹ Les tableaux de l'expérimentation sont divisés en trois sections : la représentation de l'énoncé en G-OWL à gauche, sa représentation en notation N3 à droite et sa représentation en langage naturel en bas. Dans la section de l'énoncé en G-OWL apparaît d'une part, la représentation graphique puis, pour fin d'information, la sémantique de l'objet graphique. Dans l'exemple du tableau 5, les objets graphiques *Grape* et *WineGrape* sont traduits en `owl:Class`.

Tableau 5: Exemple d'une classe nommée simple, d'un individu et la distribution hiérarchique

<pre> graph TD FoodGrape["<<food>>Grape"] WineGrape["WineGrape"] CabernetSauvignonGrape["CabernetSauvignonGrape"] FoodGrape -- "rdfs:subClassOf" --> WineGrape CabernetSauvignonGrape -- "rdf:type" --> WineGrape CabernetSauvignonGrape -- "rdfs:subClassOf" --> WineGrape </pre>	<pre> :CabernetSauvignonGrape rdf:type :WineGrape , :Grape. food:Grape rdf:type owl:Class . :WineGrape rdf:type owl:Class ; rdfs:subClassOf food:Grape . </pre>
<p><i>WineGrape</i> est une sorte de <i>Grape</i> du domaine <i>food</i>. Le <i>CabernetSauvignonGrape</i> est un <i>WineGrape</i></p>	

En OWL, la propriété peut servir à mettre en relation les individus appartenant à deux classes. Le tableau 6 présente la définition des propriétés *hasWineDescriptor* et *hasColor* qui sont identifiées par le `gowl:Attribut`.

Tableau 6 : Définition de propriétés

<pre> graph TD Wine["Wine"] hasWineDescriptor["hasWineDescriptor"] WineDescriptor["WineDescriptor"] hasColor["hasColor"] WineColor["WineColor"] Wine -- "rdfs:subPropertyOf" --> hasWineDescriptor Wine -- "rdfs:domain" --> hasWineDescriptor Wine -- "rdfs:subClassOf" --> WineDescriptor hasWineDescriptor -- "rdfs:subPropertyOf" --> hasColor hasWineDescriptor -- "rdfs:range" --> WineColor hasColor -- "rdfs:subPropertyOf" --> hasWineDescriptor </pre>	<pre> :Wine rdf:type owl:Class . :WineColor rdf:type owl:Class ; rdfs:subClassOf :WineDescriptor . :WineDescriptor rdf:type owl:Class . :hasWineDescriptor rdf:type owl:ObjectProperty ; rdfs:domain :Wine ; rdfs:range :WineDescriptor . :hasColor rdf:type owl:ObjectProperty ; rdfs:range :WineColor ; rdfs:subPropertyOf :hasWineDescriptor . </pre>
<p><i>WineColor</i> est une sorte de <i>WineDescriptor</i> la propriété <i>hasColor</i> est une sorte de propriété <i>hasWineDescriptor</i> <i>Wine hasWineDescriptor WineDescriptor</i> <i>Wine hasColor WineColor</i></p>	

3.2 Restrictions sur les propriétés

Pour représenter une restriction, le langage G-OWL utilise un *conteneur* pour agglomérer les diverses entités nécessaires. Dans le cas des restrictions *universelle* et *existentielle* (voir le tableau 7), le conteneur agglomère trois éléments soit : un type de restriction (\forall ou \exists), un et un seul `gowl:Attribut` ainsi qu'un et un seul `gowl:Concept`. En OWL, la restriction est une classe anonyme qui doit être associée à une classe nommée par la propriété `rdfs:subClassOf` ou par la propriété `rdfs:equivalentClass`. Lors de la traduction en OWL, le conteneur sera traduit en classe anonyme de type `owl:Restriction`. De même, en se référant à l'exemple du tableau 7, la restriction sera associée à la propriété *hasMaker* par la propriété `owl:onProperty` ainsi qu'à la classe *Winery* par la propriété `owl:someValuesFrom` ou `owl:allValuesFrom` selon le type de restriction choisi.

Tableau 7 : Restriction universelle et existentielle de propriétés

	<pre> :PotableLiquid rdf:type owl:Class . :Wine rdf:type owl:Class ; rdfs:subClassOf :PotableLiquid ; owl:equivalentClass [rdf:type owl:Restriction ; owl:onProperty :hasMaker ; owl:someValuesFrom :Winery] ; rdfs:subClassOf [rdf:type owl:Restriction ; owl:allValuesFrom :Winery ; owl:onProperty :hasMaker] . </pre>
<p>- <i>Wine</i> est une sorte de <i>PotableLiquid</i></p> <p>- Pour tous les <i>Wine</i>, s'ils sont associés à un producteur (<i>hasMaker</i>) alors chaque (\forall) producteur est une sorte d'Etablissement Vinicole (<i>Winery</i>)</p> <p>- Pour tous les <i>Wine</i>, il y a au moins un (\exists) producteur (<i>hasMaker</i>) qui est une sorte d'Etablissement Vinicole (<i>Winery</i>)</p>	

Au sujet des restrictions de cardinalité et de valeur (voir le tableau 8), le conteneur agglomère un type de restriction associé à la cardinalité ($=$, \leq , \geq) ou encore, associé à la représentation d'une valeur (\exists). Dans ces cas, le conteneur agglomère aussi un `owl:Attribut` ainsi qu'un `owl:Fait`.

Dans le cas d'une restriction de cardinalité, le `owl:Fait` est stéréotypé avec la mention « int » pour indiquer que la valeur représentée dans le `owl:Fait` est un `rdfs:datatype` de type `xsd:integer`. Cette mention permettra de désambigüiser *hasVintageYear* en `owl:DatatypeProperty`

Tableau 8 : Restriction de cardinalité et de valeur

	<pre> :Vintage rdf:type owl:Class ; rdfs:subClassOf [rdf:type owl:Restriction ; owl:cardinality 1^xsd:nonNegativeInteger ; owl:onProperty :hasVintageYear] . :Burgundy rdf:type owl:Class ; rdfs:subClassOf [rdf:type owl:Restriction ; owl:hasValue :Dry ; owl:onProperty :hasSugar] . :hasSugar rdf:type owl:ObjectProperty . :hasVintageYear rdf:type owl:DatatypeProperty . :Dry rdf:type owl:Thing . </pre>
<p>- Pour tous les <i>Vintage</i>, il existe un et un seul prédicat <i>hasVintageYear</i></p> <p>- Pour tous les <i>Burgundy</i>, il existe au moins un prédicat <i>hasSugar</i> associé à l'individu <i>Dry</i></p>	

3.3 Collection

La collection (`owl:Collection`) est utilisée pour agglomérer un ensemble fini de `owl:Concept`, de `owl:Restriction` ou encore un ensemble de `owl:Fait` (voir le tableau 9 et le tableau 10). La collection est entre autres utilisée pour représenter l'intersection de classes (`owl:intersectionOf`) et l'union de classes (`owl:unionOf`).

Tableau 9 : Collection

<p>owl:unionOf</p> <p>owl:Restriction</p> <p>WineDescriptor</p> <p>WineTaste</p> <p>WineColor</p> <p>WhiteWine</p> <p>Wine</p> <p>hasColor</p> <p>White</p> <p>Burgundy</p> <p>locatedIn</p> <p>BourgogneRegion</p> <p>owl:intersectionOf</p>	<pre> :WineDescriptor rdf:type owl:Class ; owl:unionOf (WineTaste WineColor) . :WhiteWine rdf:type owl:Class ; owl:intersectionOf (:Wine [rdf:type owl:Restriction ; owl:hasValue :White ; owl:onProperty :hasColor]) . :Wine rdf:type owl:Class . :White rdf:type owl:Thing . :hasColor rdf:type owl:ObjectProperty . :Burgundy rdf:type owl:Class ; owl:intersectionOf (:Wine [rdf:type owl:Restriction ; owl:hasValue :BourgogneRegion ; owl:onProperty :hasColor]) . :BourgogneRegion rdf:type owl:Thing . :WhiteBurgundy rdf:type owl:Class ; owl:intersectionOf (:Burgundy :WhiteWine) . :locatedIn rdf:type owl:ObjectProperty . </pre>
---	--

3.4 Énumération et disjonction

Le tableau 10 présente quelques exemples d'utilisation de l'énumération et de la disjonction. L'énumération, qui est en fait une sorte de collection d'individus, est utilisée pour construire un ensemble d'individus (`owl:oneOf`) ou pour distinguer un ensemble d'individus les uns des autres (`owl:distinctMembers`). Quant au `owl:LienD`, il est utilisé pour indiquer que les faits d'un concept ne peuvent pas appartenir au concept pointé par le lien. Dans l'exemple, les faits du concept *Pasta* ne peuvent pas être des faits de la classe *Meat*. Le `owl:LienD`, est traduit en OWL par `owl:disjointWith`.

Tableau 10 : Énumération et disjonction

<p>owl:oneOf</p> <p>WineColor</p> <p>White</p> <p>Red</p> <p>Rose</p> <p>EdibleThing</p> <p>Meat</p> <p>Fowl</p> <p>Pasta</p> <p>owl:disjointWith</p> <p>owl:AllDifferent</p> <p>White</p> <p>Red</p> <p>Rose</p> <p>owl:distinctMembers</p>	<pre> :WineColor rdf:type owl:Class ; owl:oneOf (:White :Red :Rose) . :EdibleThing rdf:type owl:Class . :Fowl rdf:type owl:Class ; rdfs:subClassOf :EdibleThing . :Meat rdf:type owl:Class ; rdfs:subClassOf :EdibleThing . :Pasta rdf:type owl:Class ; rdfs:subClassOf :EdibleThing ; owl:disjointWith Meat , Fowl . owl:AllDifferent owl:distinctMembers (:White :Red :Rose) . :Red rdf:type :WineColor . :Rose rdf:type :WineColor . :White rdf:type :WineColor . </pre>
--	---

4 Résultat

Pour mettre en évidence la qualité représentationnelle de G-OWL, nous avons choisis, à partir d'un fragment de l'ontologie *wine.owl*, de comparer la représentation du fragment dans sa représentation en N3, dans sa représentation en graphe et dans sa représentation en G-OWL (voir la fig. 2).

La représentation de comparaison de type graphe a été choisi puisqu'il s'agit d'une représentation couramment employée par les éditeurs d'ontologie tel que *Protégé* et *TopBraid Composer*. De plus, cette représentation en graphe est l'une des seules représentations qui permet de représenter des individus, des classes, des axiomes et des propriétés dans un même diagramme.

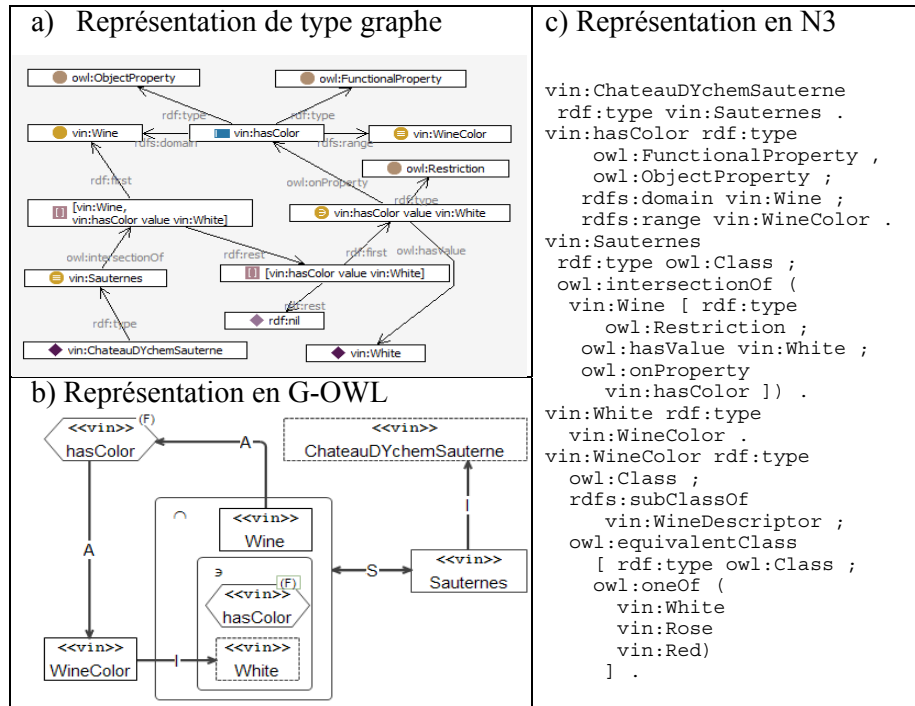


Fig. 2 –Ontologie partielle du Sauterne Château d'Yquem en représentation graph, G-OWL et N3, extrait de *wine.owl* (Deborah L. McGuinness, *et al.*, 2004)

Le fragment de l'ontologie présentée à la fig. 2 représente l'idée qu'un Château d'Ychem Sauterne est un vin blanc de Sauternes. Pour représenter cette idée, il est inscrit dans l'ontologie que l'individu *ChateauDYchemSauterne* est de type *Sauternes*. Le *Sauternes* est la conjonction d'un *Wine* et de quelque chose qui est de couleur (*hasColor*)

White. La propriété `hasColor` a pour domaine `Wine` et à pour co-domaine (le `rdfs:range`) `WineColor`. White est une instance de `WineColor`.

Pour l'évaluation, le tableau 11 présente le dénombrement des différents éléments graphiques de chaque représentation. On constate que pour chaque critère, G-OWL comporte un nombre d'éléments inférieur au nombre d'éléments du graphe et ceci, qu'il s'agisse du nombre de type d'éléments ou du nombre total d'éléments. Sans être une garantie formelle de simplicité d'usage, ces indicateurs laissent présager une tendance vers une simplification et une convivialité de l'usage de G-OWL.

Tableau 11 : Table de comparaison des représentations graphiques graphe et G-OWL

Critère	Graphe	G-OWL
Nombre de sorte d'entités	6	5
Nombre de sorte de relations	8	3
Nombre total d'entités	13	9
Nombre total de relations	13	5

Au niveau de l'intuitivité, l'usage de relations de type `rdf:rest`, `rdf:first` nécessaire à la construction grammaticale de l'ontologie, est en revanche contre-productif pour un expert de contenu qui désire interpréter l'ontologie au vue de son domaine du discours. Ces restrictions grammaticales tournées vers l'exploitation automatique d'une ontologie OWL se trouve donc camouflé en G-OWL, qui lui, s'attarde sur l'utilisation de symbole signifiants en termes de représentation d'un domaine de discours.

5 Conclusion

Cet article a présenté G-OWL, un langage graphique pour la construction d'une ontologie dans le formalisme OWL qui repose sur l'hypothèse qu'un langage graphique, polysémique, polymorphique et typé offre une convivialité d'utilisation accrue par rapport à un langage formel textuel comme l'OWL ou encore un langage graphique n'utilisant pas de polysémie comme le graphe ou l'UML. Le vocabulaire graphique de G-OWL ont été définis et mis en contexte d'utilisation par plusieurs exemples illustrant l'expressivité d'OWL. La présentation de la modélisation graphique partielle de l'ontologie *Wine.owl* démontre qu'il est possible de représenter une ontologie relativement complexe, contenant plusieurs éléments d'expressivité d'OWL dans un seul schéma qui en synthétise la représentation. La comparaison avec une représentation en graphe a démontré que G-OWL utilise moins d'entités, de relations et de sorte d'entités et de sorte de relations. Cette diminution dans l'usage des symboles combinée à l'hypothèse (démontrée dans des travaux de recherches antérieures) que la simplicité d'usage d'un langage

est liée à la limitation du nombre de symboles nécessaire pour soutenir une certaine expressivité, laisse fortement présager que G-OWL est plus simple d'usage qu'un langage non polysémique.

Bien que G-OWL soit à une étape avancée de la définition de ses spécifications, il est nécessaire de poursuivre sa validation afin d'en assurer la cohérence, la complétude et la convivialité d'usage. G-OWL doit donc être utilisé pour construire d'autres ontologies et il doit être mis à l'essai sur le terrain lors de séances d'élicitation et de remue-méninge. De plus, la cohérence et la complétude du langage pourront être formellement validées par la construction d'un logiciel d'édition d'ontologies en G-OWL. Le caractère systématique et formel du processus d'implantation d'un tel éditeur permettra de relever plusieurs points d'incohérence et de non-complétude du langage. Pour ce qui est de la validation d'usage, il serait nécessaire, après une formation adéquate sur le langage G-OWL, de demander à des experts de contenu de construire des ontologies de leur domaine avec G-OWL. Cette validation d'usage permettra de mesurer la capacité de G-OWL à être utilisé de manière intuitive.

Grâce à son vocabulaire polysémique exprimé de manière graphique, polymorphique et typé, qui allège le processus mental de conception, nous pensons que G-OWL pourrait-être utilisé par des experts de contenus pour l'élicitation de connaissances et la production d'ontologies qui pourraient être validé « à chaud » si l'éditeur G-OWL est couplé avec un moteur d'inférence. Les conclusions produites par le moteur d'inférence pourraient être présentées directement et de manière graphique à l'expert lui permettant de valider la représentation de la connaissance qu'il se fait de son domaines du discours. G-OWL pourrait être utilisé, par exemple, dans des activités de transferts de connaissances ou pour la construction de bases de connaissances de systèmes experts dans une représentation accessible à l'expert de contenu.

6 Références

- Baader, F., Calvanese, D., McGuinness, D. L., Nardi, D., & Patel-Schneider, P. F. (2007). *The Description Logic Handbook*. Cambridge University Press.
- Basque, J., & Pudelko, B. (2010). Modeling for Learning. In G. Paquette (Ed.), *Visual Knowledge and Competency Modeling - From Informal Learning Models to Semantic Web Ontologies*. Hershey, New York: IGI Global.
- Buzan, T., & Buzan, B. (1994). *The Mind Map Book: How to Use Radiant Thinking to Maximize Your Brain's Untapped Potential*. E P Dutton.
- Chein, M., & Mugnier, M.-L. (2008). *Graph-based Knowledge Representation*. London: Springer.
- Eskridge, T., Hayes, P., & Hoffman, R. (2006). *Formalizing The Informal: A Confluence Of Concept Mapping And The Semantic Web*. Paper presented at the Second Int. Conference on Concept Mapping San José, Costa Rica.

- Gašević, D., Djurić, D., & Devedžić, V. (2006). *Model Driven Architecture and Ontology Development*. New York, Inc.: Springer-Verlag.
- Héon, M. (2010). *OntoCASE: Méthodologie et assistant logiciel pour une ingénierie ontologique fondée sur la transformation d'un modèle semi-formel*. Unpublished Thèse, Université du Québec à Montréal, Montréal.
- Héon, M. (2012). OntoCASE un outil CASE d'élucitation et de conception d'une ontologie OWL. Retrieved 20 février 2012, 2012, from <http://www.cotechnoe.com/ontocase>
- Kinshuk, H. H. A., Pawlowski, J. M., & Sampson, D. (2008). *Handbook for Education and Training on Information Technologies* (Second edition ed.). Heidelberg: Springer-Verlag.
- McGuinness, D. L., & Harmelen, F. v. (2004). OWL Web Ontology Language Overview Retrieved 30 mai, 2006, from <http://www.w3.org/TR/owl-features/>
- McGuinness, D. L., Patel-Schneider, P. F., Thomason, R. H., Abrahams, M. K., Resnick, L. A., Cavalli-Sforza, V., et al. (2004). Wine Ontology, OWL Web Ontology Language Guide. 2011, from <http://www.w3.org/TR/2004/REC-owl-guide-20040210/wine.rdf>
- Novak, J. D., & Cañas, A. J. (2006). The origins of the concept mapping tool and the continuing evolution of the tool. *Information Visualization*, 5, 175–184.
- Paquette, G. (2002). *L'ingénierie pédagogique*. Sainte-Foy (Québec): Presses de l'Université du Québec.
- Paquette, G. (2010). *Visual Knowledge and Competency Modeling - From Informal Learning Models to Semantic Web Ontologies*. Hershey, PA: IGI Global.
- Rumbaugh, J., Jackson, I., & Booch, G. (1999). *The Unified Modeling Language Reference Manual*.
- Sowa, J. F. (1999). Conceptual Graph. In P. Bernus, G. Schmidt & K. Mertins (Eds.), *Handbook on Architectures of Information Systems* (pp. 834): Springer-Verlag New York, Inc.
- Yang, Y. (2006). EMF Ontology Definition Metamodel. 2008, from <http://wiki.eclipse.org/MDT-EODM>
- Yu, L. (2011). OWL: Web Ontology Language. In Springer (Ed.), *A Developer's Guide to the Semantic Web* (pp. 155-239): Springer Berlin Heidelberg.