Un métamodèle des graphes conceptuels

Olivier Gerbé* — **Guy W. Mineau**** — **Rudolf K. Keller*****

** HEC Montréal*
3000, chemin de la Côte-Sainte-Catherine, Montréal, Québec, Canada H3T 2A7
olivier.gerbe@hec.ca

*** Université Laval*
Québec, Québec, Canada G1K 7P4
mineau@ift.ulaval.ca

**** Université de Montréal*
C.P. 6128 Succursale Centre-Ville, Montréal, Québec, Canada H3C 3J7
keller@IRO.UMontreal.ca

RÉSUMÉ. *La métamodélisation consiste à définir formellement le vocabulaire et les règles utilisés dans les activités de modélisation. Elle est souvent identifiée comme une composante clé dans le développement de systèmes d'information parce qu'elle définit d'une manière formelle les primitives de modélisation qui seront utilisées dans les activités de conception et d'analyse des systèmes d'information. Les graphes conceptuels sont souvent identifiés comme un langage clé dans la représentation de la connaissance de par leur simplicité, leur expressivité et leur similitude à d'autres langages de modélisation graphiques tels UML ou Entité-Relation. Nous proposons dans cet article un métamodèle des graphes conceptuels.*

ABSTRACT. *Metamodeling addresses the issue of formally defining the vocabulary and rules subsequently used in modeling activities. On one hand, metamodeling is often identified as a key component in information system development. It defines in a formal way the modeling primitives that will be used in system analysis and design activites. On other hand, conceptual graphs are often identified as a key language in knowledge representation because they are simple, expressive and closelyrelated to other familiar modelling languages such as UML or E/R. We propose in this paper a metamodel for conceptual graphs.*

MOTS-CLÉS : *métamodélisation. graphes conceptuels, représentation de la connaissance.*

KEYWORDS: *metamodelling, conceptual graphs, knowledge representation.*

RSTI - RIA. Volume 21 – n° 2/2007, pages 257 à 286

1. Introduction

La métamodélisation, qui consiste à définir formellement le vocabulaire et les règles utilisés dans les activités de modélisation, est une technique de plus en plus utilisée en modélisation dans différents domaines (Bernstein *et al.*, 2000; Flatscher, 2002) car cette formalisation du langage facilite la modélisation et évite les différentes interprétations des modèles développés. L'approche d'ingénierie dirigée par les modèles (Favre *et al.*, 2006) a renforcé l'intérêt pour la métamodélisation.

En définissant le langage de modélisation, la métamodélisation permet d'intégrer les contraintes sémantiques dans le langage et d'ainsi contraindre l'expressivité du langage et d'assurer ainsi une meilleure cohérence. Cette formalisation permet également d'interroger et de manipuler les modèles. De plus, la définition formelle du langage de modélisation nous donne les outils pour expliquer et fournir de l'aide pour la compréhension de ses éléments. Ceci garantit une utilisation uniforme et cohérente du langage parmi les utilisateurs.

Les applications des techniques de métamodélisation sont nombreuses. Dans le monde du génie logiciel, la métamodélisation est au cœur de l'ingénierie dirigée par les modèles qui permet de concevoir des modèles indépendants des plates-formes d'implémentation (OFTA, 2004; OMG, 2006). Dans le domaine des bases de données, la métamodélisation est utilisée dans la gestion de méta-données (Bernstein *et al.*, 2000). Dans les systèmes de diffusion multimédias, la métamodélisation offre des moyens de mettre en œuvre la qualité de service par la modélisation des besoins des usagers et des fonctionnalités des systèmes utilisés (Kerherve *et al.*, 2006). Dans le cadre du Web sémantique, la métamodélisation et l'utilisation des graphes conceptuels pourraient être un atout important pour l'interopérabilité des systèmes et des travaux (Berners-Lee, 2001; Delteil *et al.*, 2001; Corby *et al.*, 2000) en témoignent.

Dans le domaine de la représentation de connaissances les graphes conceptuels sont souvent identifiés comme un langage clé de par leur simplicité, leur expressivité et leur similitude à d'autres langages de modélisation graphiques bien connus tels UML ou Entité-Relation. Bien que la notation des graphes conceptuels soit en cours de standardisation (ISO, 2005) la métamodélisation des graphes conceptuels n'a été que très peu étudiée. Esch en 1994 (Esch, 1994a) et Wermelinger en 1995 (Wermelinger, 1995) proposent les premières discussions et formalisations du langage des graphes conceptuels en ce sens. Depuis cette date, peu de travaux se sont intéressés à la modélisation des graphes conceptuels : quelques travaux sur les graphes conceptuels et RDF (Resource Description Framework) (Berners-Lee, 2001; Delteil *et al.*, 2001; Corby *et al.*, 2000; Gerbé *et al.*, 2002) et sur la métamodélisation (Gerbé *et al.*, 2003b).

Pour combler cette lacune, nous proposons ici un métamodèle pour les graphes conceptuels. L'intérêt des graphes conceptuels réside principalement dans le fait qu'ils sont un des rares formalismes où la définition des types se fait en donnant un exemple. Pour définir un chauffeur de taxi, nous exprimons le fait qu'un chauffeur de taxi est une personne qui conduit un taxi. Ce fait complique un peu l'écriture de définition des types, comme nous le verrons plus loin, mais permet d'éviter de développer des méca-

nismes externes de contrôle pour vérifier que les données entrées sont bien conformes à leur définition.

L'article est donc organisé comme suit. La section 2 expose le formalisme des graphes conceptuels avec ses principes et ses opérateurs tels que définis par la communauté des graphes conceptuels. La section 3 présente notre proposition de métamodèle et la section 4 détaille d'une manière exhaustive les composants du métamodèle des graphes conceptuels en utilisant les graphes conceptuels eux-mêmes. Enfin la section 5 conclut et présente les futurs axes de recherche.

2. Les graphes conceptuels

Cette section est une présentation des graphes conceptuels. Sowa a introduit dans (Sowa, 1984) la première version du formalisme des graphes conceptuels. Depuis de nombreux travaux ont porté sur cette formalisation ainsi que sur ses extensions (Chein *et al.*, 1992; Mugnier, 1995; Mugnier *et al.*, 1996; Wermelinger, 1995). Mugnier et Chein (Chein *et al.*, 1992; Mugnier *et al.*, 1996; Baget *et al.*, 2002; Chein *et al.*, 2004) ont généralisé le modèle de base de Sowa (Sowa, 1984) en levant certaines restrictions (connexité des graphes, treillis de types). Le langage présenté ici s'appuie sur ces différents travaux et sur le standard CGIF (Conceptual graph Interchange Format) (ISO, 2005) en cours d'élaboration. Cependant toutes les utilisations du langage ne sont pas définies explicitement. C'est pourquoi, en nous fondant sur la terminologie employée dans (Mugnier *et al.*, 1996), nous donnons ci-dessous les principes fondamentaux et les opérations de base sur les graphes conceptuels.

2.1. *Introduction*

Le lecteur peu familier avec les graphes conceptuels trouvera ici une brève introduction au formalisme ; seulement le minimum requis pour la compréhension de la suite de cet article est présenté. Plus d'information peut être trouvée dans (Sowa, 1984; Chein *et al.*, 1992; Sowa, 2000).

Les graphes conceptuels sont un formalisme par lequel l'univers du discours est représenté par des concepts et des relations conceptuelles. Un concept représente un objet du discours. Les relations conceptuelles permettent d'associer les concepts. La figure 1 représente la phrase « John est un chauffeur de taxi qui conduit le taxi numéro 23 ». La même phrase peut être représentée sous une forme linéaire ou sous une forme graphique.

```
[ChauffeurTaxi :John]->(conduit)->[Taxi :#23].
```

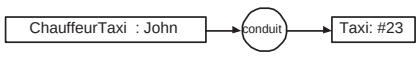

Figure 1. *Graphes conceptuels : concepts et relations*

Les concepts peuvent être catégorisés par les relations qu'ils entretiennent avec les autres concepts. Les types de concepts appelés `Concept Type` définissent ces catégories. Un type de concept est défini par un graphe de définition. La figure 2 présente le graphe de définition du type `ChauffeurTaxi` qui spécifie qu'un chauffeur de taxi est un conducteur qui conduit un taxi. De la même façon, les relations peuvent être catégorisées par les concepts qu'elles lient. Les types de relation sont appelés `Relation Type`. Un type de relation est également défini par un graphe. La figure 2 présente le graphe de définition du type `mange`.

```
Type ChauffeurTaxi(x) is
    [Chauffeur :x]->(conduit)->[Taxi]
Type mange(x,y) is
    [EtreAnime :x]->(mange)->[Nourriture :y]
```

Figure 2. *Graphes conceptuels : Type de concept et type de relation.*

2.2. *Principes*

Nous rappelons ici les principes fondamentaux qui régissent le formalisme des graphes conceptuels.

Principe 1 *Les graphes ne sont pas nécessairement connexes.*

Principe 2 *L'ensemble des types de concepts forme un sous-ensemble du treillis engendré par la fermeture des types de concepts par les opérations de conjonction (intersection) et disjonction (union).*

Principe 3 *Les graphes sont sous forme normale (Mugnier* et al.*, 1996). Deux concepts identiques, c'est à dire, représentant la même entité, sont systématiquement joints. Deux graphes ou parties de graphes ayant deux concepts identiques sont fusionnés pour n'en faire qu'un seul ce qui implique aussi une étape de simplification où toute relation dupliquée est systématiquement éliminée du nouveau graphe.*

2.3. *Opérations de base sur les graphes conceptuels*

Nous rappelons ici les opérations et définitions de base définies dans (Sowa, 1984) pour la dérivation des graphes conceptuels. La dérivation permet de créer un nouveau graphe à partir de graphes existants.

Soient w le graphe dérivé de u et v (u et v pouvant être identiques). Il y a quatre opérations de dérivation :

Simplification.

Si deux relations conceptuelles sont identiques, c'est à dire si elles ont même arité, même ordre des concepts liés et même type, une de deux relations peut être supprimée ainsi que les arcs la reliant aux concepts qu'elle associe.

Restriction de concept.

Un concept de u peut être restreint en remplaçant son type par un type plus spécialisé (sous-type) ou en remplaçant un concept générique représentant une entité anonyme (inconnue) par un concept individuel représentant une entité identifiée (connue).

Restriction de relation.

Le type d'une relation de u peut être remplacé par un sous-type.

Jointure.

Si un concept c de u est identique à un concept d de v, w est le graphe résultant de l'union de u et v en supprimant d et en reliant les arcs arrivant et partant de d à c.

Nous rappelons maintenant deux définitions et un théorème qui précisent comment les graphes conceptuels peuvent être dérivés.

Définition 1 *Un graphe w est dit canoniquement dérivable d'un ensemble de graphes $\mathcal{A}$ si une des deux conditions suivantes est remplie :*

– w est un élément de $\mathcal{A}$;

– w est un graphe dérivé de graphes eux-mêmes canoniquement dérivables de $\mathcal{A}$.

Définition 2 *Si un graphe u est canoniquement dérivable d'un graphe v, u est dit une spécialisation de v et noté $u \leq v$, et v est dit une généralisation de u.*

Théorème 1 *La généralisation définit un ordre partiel sur les graphes conceptuels, appelé hiérarchie de généralisation. Pour tout graphe u, v et w, on a les propriétés suivantes :*

– réflexivité : $u \leq u$;

– transitivité : si $u \leq v$ et $v \leq w$ alors $u \leq w$;

– antisymétrie : si $u \leq v$ et $v \leq u$ alors $u = v$;

– subsomption : si v est un sous-graphe de u alors $u \leq v$;

– sous-typage : si u est identique à v excepté que des concepts de v ont été restreints à des sous-types dans u alors $u \leq v$;

Enfin nous rappelons l'opération fondamentale des graphes conceptuels appelée projection qui permet le calcul de la relation de spécialisation. Cette opération est à la base du mécanisme de recherche dans les graphes conceptuels.

Théorème 2 *Pour tous graphes conceptuels u et v tels que $u \leq v$, il existe une correspondance $\pi : v \rightarrow u$, où $\pi(v)$ est un sous-graphe de u appelé la projection de v dans u. L'opération de projection a les propriétés suivantes :*

– Pour tout concept c de v, $\pi(c)$ est un concept de $\pi(v)$ et le type de $\pi(c)$ est le même que celui de c ou en est un sous-type ;

– Pour toute relation conceptuelle r de v, $\pi(r)$ est une relation conceptuelle de $\pi(v)$ et le type de $\pi(r)$ est le même ou est un sous-type du type de r. De plus si le $i^{ème}$ arc de r est connecté à un concept c dans v, alors le $i^{ème}$ arc de $\pi(r)$ est connecté à $\pi(c)$ dans $\pi(v)$.

Le lecteur intéressé trouvera les démonstrations des théorèmes présentés ci-dessus dans (Sowa, 1984).

3. Proposition d'un métamodèle

Excepté les travaux en cours pour un standard d'un format d'échange (ISO, 2005), les travaux portant sur la description formelle d'un métamodèle des graphes conceptuels sont très réduits, voire inexistants. John Esch (Esch, 1994b) effleure le sujet et propose l'introduction de deux relations : Kind qui lie un concept au type dont il est instance et Subt qui lie deux types tels que l'un est sous-type de l'autre. Il définit grâce à la relation Subt des hiérarchies de types pour chacun des niveaux et relie les éléments des hiérarchies par la relation Kind. Esch n'aborde pas le problème des relations et types de relation. Pour sa part, Michel Wermelinger définit de façon plus formelle les types d'ordre supérieur et propose une traduction vers la logique du premier ordre (Wermelinger, 1995). Il considère qu'il existe une hiérarchie pour les types de concepts et une hiérarchie pour les types de relations. Wermelinger classifie les types de concepts suivant leur nature et leur ordre. Il distingue les types de concepts relationnels dont les instances sont des types de relations et les autres types de concepts dont les instances sont des concepts ou des types de concepts. Les types de relations sont classifiés en fonction de leur arité et de leur ordre. Les instances de types de relations sont des relations.

Ces deux approches sont compatibles et le travail présenté ici étend certaines notions introduites par ces auteurs. Comme eux, nous définissons deux hiérarchies de types, une pour les types de concepts et une pour les types de relations.

Dans cette section, nous présentons d'abord des principes additionnels que nous ajoutons au formalisme dans le but de contrôler l'acquisition de connaissances, puis nous introduisons notre proposition de hiérarchie des types (concepts et relations) du métamodèle et décrivons sommairement chacun de ces types. Enfin nous présentons la notation utilisée pour la description du métamodèle.

3.1. *Principes additionnels*

Nous ajoutons trois principes qui régissent notre métamodèle. Le premier principe statue que l'opération de projection est injective, donc si deux concepts sont distincts alors ils ne représentent pas le même objet du discours. Les deux autres principes permettent de contrôler l'acquisition de connaissances. Ils statuent que l'acquisition se fait conformément aux spécifications déjà définies dans la base. Par exemple, un concept ne peut être saisi que si le type du concept a été préalablement défini dans la base, de même pour les relations.

Principe 4 *L'opération de projection est injective. Si deux concepts sont distincts alors ils se projettent sur des concepts distincts.*

Principe 5 *Tout concept doit vérifier le graphe de définition de son type. Une relation n'existe qu'en relation avec un type de concept. Pour qu'une relation puisse être établie entre deux concepts, il faut nécessairement que cette relation apparaisse dans au moins un des graphes de définition des types des concepts concernés.*

Principe 6 *Seules les propositions réputées vraies sont insérées dans la base de connaissance. Toute proposition identifiée comme fausse, c'est à dire, qui n'est pas conforme aux graphes de définition des types est rejetée au moment de l'acquisition des connaissances.*

3.2. *Vue d'ensemble*

Nous présentons ici l'ensemble des types (concepts et relations) du formalisme que nous proposons. Ces types seront formellement définis et détaillés à la section 4.

La figure 3 illustre la hiérarchie des types que l'on retrouve dans le métamodèle[1].

Au plus haut niveau de la hiérarchie nous distinguons deux catégories, les éléments externes (`ExternalElt`), et les éléments internes (`InternalElt`) qui sont les constituants du langage. Les éléments externes représentent les objets de l'univers du discours qui sont à l'extérieur du système et qui peuvent être référencés par des éléments internes. Les éléments internes du langage sont regroupés en six types : référent, graphe, contexte, type, lien de coréférence et élément de graphe. Les référents (`Referent`) sont les représentants des objets de l'univers du discours ; les graphes (`Graph`) sont les phrases du langage, les contextes (`Context`) regroupent les graphes représentant la connaissance, les types (`Type`) permettent de regrouper les référents en catégories, les liens de coréférence (`CoRefLink`) lient les concepts représentant les

1. Les libellés des types de concept sont en anglais pour des raisons d'intégration avec le standard en cours de définition. Cependant le lecteur trouvera un libellé français de chacun des types de concepts lorsqu'ils seront définis à la section 4.

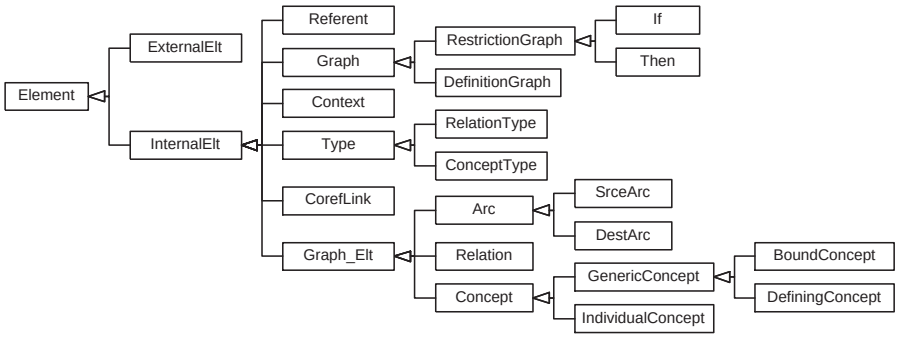

Figure 3. *Hiérarchie des types de concepts du langage*

mêmes éléments et les éléments de graphes (`Graph_Elt`) sont les éléments : arc (`Arc`), relation (`Relation`) ou concept (`Concept`), que l'on retrouve dans un graphe.

Parmi les concepts on distingue les concepts individuels (`IndividualConcept`) qui représentent des entités identifiées de l'univers du discours et les concepts génériques (`GenericConcept`) qui représentent des entités non identifiées.

Parmi les graphes nous distinguons deux sous-types : définition et restriction. Les graphes de définition `DefinitionGraph` sont des graphes utilisés dans la définition des types de concept et des types de relation. Les graphes de restriction `RestrictionGraph` sont des graphes qui doivent être toujours faux et qui contraignent les définitions de types de concept. Les graphes `If` et `Then` sont des cas particuliers de graphes de restriction.

Les figures 4 et 5 montrent les liens qui existent entre ces types en utilisant le formalisme UML. La figure 4 présente les éléments qui constituent un graphe. Un graphe est constitué de concepts, de relations et d'arcs source et destination qui lient les concepts et les relations. Un concept est composé de deux parties : un type de concept, lié à d'autres types de concepts par la relation de sous-typage, et un référent qui représente un et un seul élément. Une relation est associée à un type de relation qui définit la sémantique de la relation conceptuelle. Enfin un contexte permet de segmenter l'information en regroupant des graphes par une relation d'extension et en spécifiant par une relation d'intention les graphes pour lesquels les graphes de l'extension sont vrais.

La figure 5 présente les éléments définissant les types, types de concepts et types de relations. Un type est spécifié par : un graphe de définition, des graphes de restriction et des graphes de règles.

La figure 6 présente la hiérarchie des types de relations définis dans le langage. La relation de définition `def` et la relation de restriction `rstrct` associent les types à leurs graphes de spécifications.

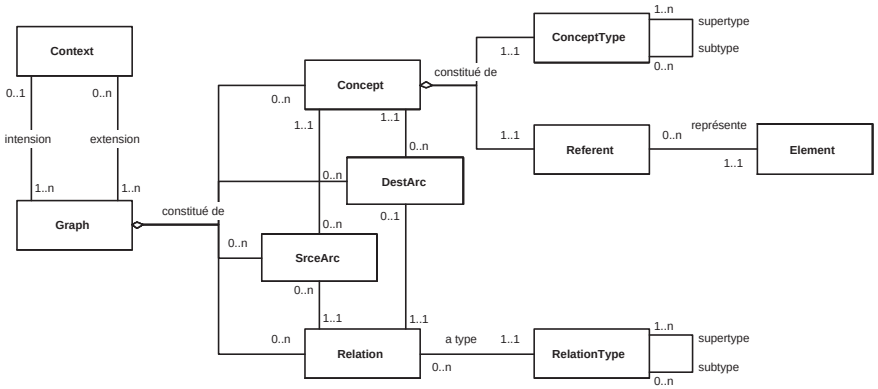

Figure 4. *Le métamodèle - partie 1(formalisme UML)*

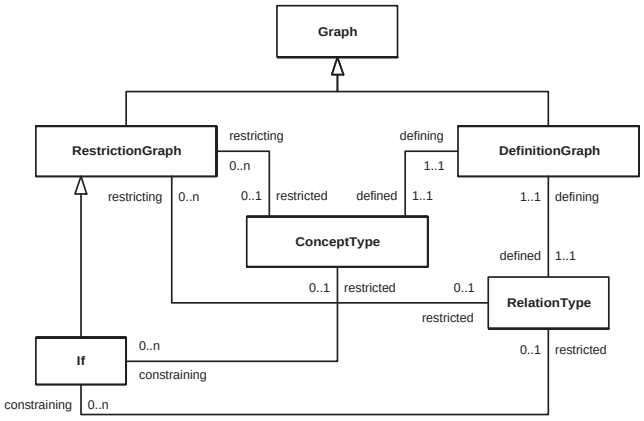

Figure 5. *Le métamodèle - partie 2 (formalisme UML)*

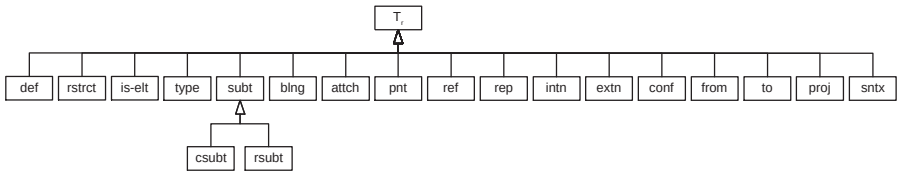

Figure 6. *Hiérarchie des types de relations du langage*

La relation d'appartenance `is-elt` associe les éléments d'un graphe au graphe. La relation `type` lie un concept (resp. relation) à son type de concept (resp. relation). La relation de sous-typage `subt`, avec ses deux spécialisations `csubt` pour les types

de concepts et `rsbut` pour les types de relations, organise les types en hiérarchie. La relation `blng` associe les arcs aux relations et la relation `attch` associe les arcs aux concepts. La relation `pnt` est une relation d'orientation pour les arcs. Les relations `ref` et `rep` associent respectivement un concept à son référent et un référent à l'objet qu'il représente. Les relations `intn` et `extn` associent un contexte aux graphes qui le définissent. La relation de conformité `conf` associe un référent à un type. Les relations `from` et `to` associent un lien de coréférence à respectivement la source du lien et la destination du lien. La relation de projection `proj` associe un graphe à un graphe plus spécialisé. Enfin la relation de syntaxe `sntx` associe un type au graphe qui spécifie la syntaxe.

3.3. *Notation et description du métamodèle*

Dans (Gerbé *et al.*, 2002) nous postulions que le formalisme des graphes conceptuels était flexible et suffisamment puissant pour être un langage pivot pour les langages de représentation dédiés au Web. Afin de confirmer cette assertion, nous décrivons le métamodèle des graphes conceptuels en utilisant les graphes conceptuels eux-mêmes.

Les éléments importants du métamodèle sont les types. Nous avons introduit trois catégories de graphes pour spécifier les types :

– graphe de définition

– graphe de restriction

– graphe de règle

comme illustré à la figure 7. Cette figure présente le graphe de spécification du type de concept Conducteur. Le graphe de spécification contient les graphes de définition, de restriction et de règles qui donnent les conditions nécessaires et suffisantes pour la reconnaissance des instances du type de concept Conducteur. Le graphe de définition est lié au type de concept par la relation def. Les graphes de restriction et leurs spécialisations les graphes de règles sont liés au type de concept par la relation rstrct.

Graphe de définition.

Le graphe de définition présente les relations conceptuelles nécessaires pour qu'un concept soit du type défini. Pour les besoins de notre exemple, un conducteur est une personne qui conduit un véhicule.

D'une manière formelle (Sowa, 1984), un type de graphe est défini par une lambda expression, l'exemple de la figure 7 correspond à l'équation suivante :

`Conducteur = [Personne : λ]→(conduit)→[Vehicule].`

Le caractère λ indique que le concept Personne est le paramètre formel. Dans la version graphique ou la forme linéaire le caractère λ est remplacé par un point d'interrogation.

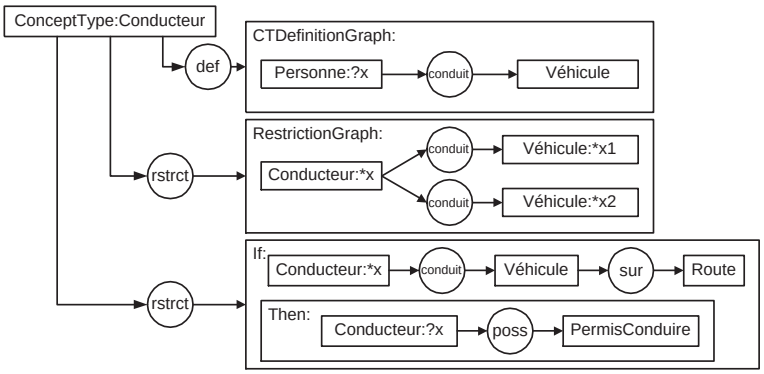

Figure 7. *Définition du type* Conducteur

Graphe de restriction.

Le ou les graphes de restriction spécifient des conditions nécessaires complémentaires ; ces graphes précisent les relations conceptuelles qui ne peuvent pas être liées au concept. Dans notre exemple, le graphe de restriction précise qu'un conducteur ne peut pas conduire deux véhicules[2].

Graphe de règle.

Le ou les graphes de règles spécifient d'autres conditions nécessaires ; ces graphes présentent les règles que doit observer un concept. Dans l'exemple, le graphe de règle exprime le fait que si un conducteur conduit un véhicule sur une route alors il doit posséder un permis de conduire.

4. Les types de concepts du métamodèle

Pour chacun des types nous donnons la définition et le graphe de spécification. Les types sont introduits dans un ordre qui, nous l'espérons, facilite la compréhension des composantes du métamodèle.

4.1. *Les éléments de base*

4.1.1. *Graphe conceptuel (Graph)*

Basés sur la logique des prédicats du premier ordre, les graphes conceptuels offrent une représentation, graphique ou linéaire, d'un domaine de connaissance.

2. Nous rappelons que les graphes sont en forme normale et que l'opération de projection est injective. Si deux concepts sont distincts alors ils se projettent sur des concepts distincts.

Définition 3 *Un graphe conceptuel est un graphe composé de deux sortes de nœuds : des concepts et des relations conceptuelles qui associent les concepts par des arcs orientés.*

La figure 8 présente les graphes qui spécifient le type Graph. Un graphe conceptuel peut être vide et ne contenir aucun concept, aucune relation conceptuelle et aucun arc. Le graphe de définition du type Graph ne contient donc qu'un seul concept qui exprime qu'un graphe est un élément interne.

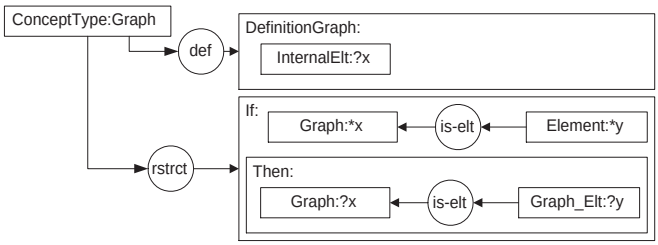

Figure 8. *Graphe de spécification de* Graph

Le graphe de règles fournit l'information sur les éléments qui peuvent être contenus dans un graphe : si un élément fait partie d'un graphe conceptuel alors cet élément est un Graph-Elt, c'est-à-dire, soit un concept, soit une relation, soit un arc. Dans ce dernier cas, la définition de Arc montre que le graphe contiendrait également un concept et une relation.

4.1.2. *Concept (Concept)*

La notion de concept est la notion fondamentale de la théorie des graphes conceptuels. Un concept représente un objet, une idée, ou tout autre notion perceptible et exprimable.

Définition 4 *Un concept est la représentation d'un objet de l'univers du discours. Il est l'assemblage de deux parties : un référent qui identifie l'objet représenté et un type qui classifie l'objet représenté.*

La figure 9 présente les graphes de définition qui spécifient le type Concept. Le premier graphe représente la définition suivante : un concept est constitué d'un type, d'un référent et est un élément d'un graphe conceptuel.

Les deux graphes de restriction précisent les règles de constitution : un concept a un et un seul référent et un et un seul type.

Le référent est dit *conforme* au type si l'objet qu'il représente est directement ou indirectement du type qu'on lui associe. Il est à noter qu'un même type peut être associé à des référents différents et constituer des concepts différents ; de la même façon un référent peut se conformer à des types différents et constituer des concepts

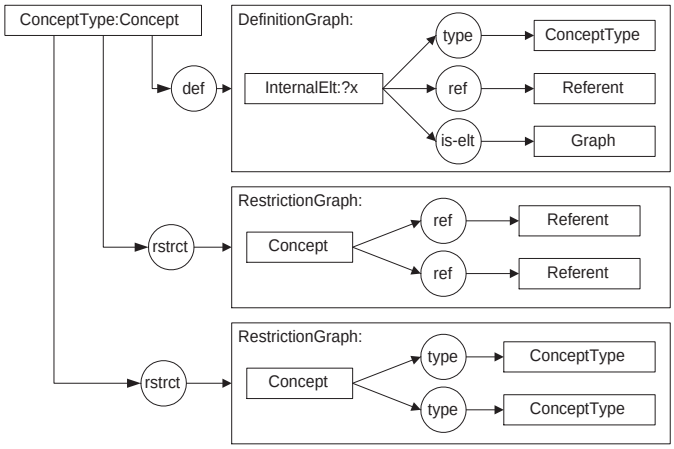

Figure 9. *Graphe de spécification de* Concept

différents. Ce mécanisme permet la représentation de points de vue : un même objet physique peut être perçu de manières différentes.

4.1.3. *Référent (Referent)*

Un référent est la partie d'un concept qui représente et identifie l'objet de l'univers du discours dont le concept est une interprétation.

Définition 5 *Un référent est un représentant d'un objet de l'univers du discours dans la base de connaissance. Un référent est symbolisé par un quantificateur et un 'désignateur' qui réfère à l'objet.*

La figure 10 présente le graphe de spécification de Referent. Un référent est un élément interne du langage qui est un constituant d'un concept et qui représente un élément (interne ou externe au langage). Un référent ne peut pas représenter deux éléments et un élément ne peut pas être représenté par deux référents.

4.1.4. *Concept générique (GenericConcept)*

Il existe deux sortes de concepts suivant que l'élément représenté par le concept est connu ou inconnu.

Définition 6 *Un concept générique est un concept dont le référent est inconnu. Le concept est une interprétation d'un objet du discours qui existe mais que l'on ne peut pas identifier.*

La figure 11 présente le graphe de spécification de GenericConcept.

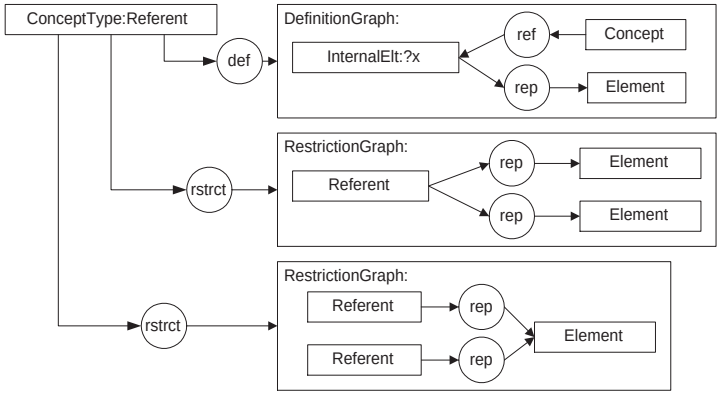

Figure 10. *Graphe de spécification de* Referent

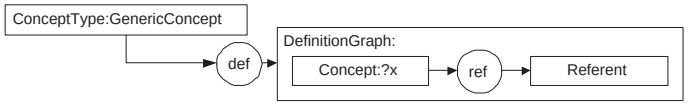

Figure 11. *Graphe de spécification de* GenericConcept

4.1.5. *Concept individuel (Individual Concept)*

Définition 7 *Un concept individuel est un concept dont le référent est connu. Le concept est une interprétation d'un objet du discours qui est identifié et représenté dans le système par le référent.*

La figure 12 présente le graphe de spécification de Individual. Un concept individuel a un référent qui représente un élément.

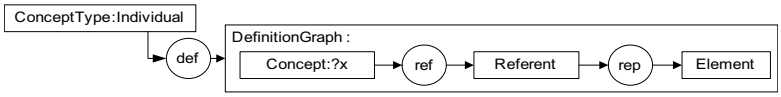

Figure 12. *Graphe de spécification de* IndividualConcept

4.2. *La coréférence*

4.2.1. *Lien de coréférence (CoRefLink)*

Les liens de coréférence permettent d'indiquer que des concepts représentent le même élément, que cet élément soit connu ou non.

Définition 8 *Un lien de coréférence est un lien orienté entre deux concepts : un concept maître et un concept lié. Le concept lié représente le même élément que le concept maître.*

La figure 13 présente le graphe de spécification de CoRefLink. Un lien de coréférence est issu d'un concept, le concept maître, et est associé à un autre concept, le concept lié. Un lien de coréférence ne peut pas être issu de deux concepts différents. Il ne peut pas non plus être associé à deux concepts liés différents.

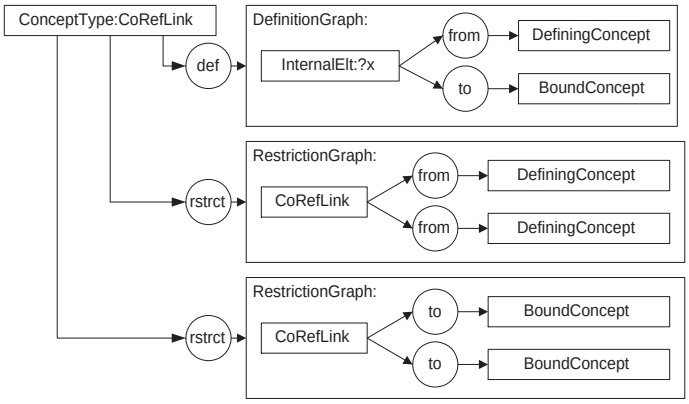

Figure 13. *Graphe de spécification de* CoRefLink

Un même concept maître peut être associé à plusieurs concepts liés. Tous les concepts liés représentent alors le même élément, celui défini par le concept maître. L'ensemble de concepts formé par un concept maître et ses concepts liés est appelé un *ensemble de coréférence*. Les liens de coréférence peuvent être représentés par des lignes pointillées ou par des libellés de coréférence dans la partie référent d'un concept. Le libellé de coréférence du concept maître est précédé d'une astérisque et le même libellé de coréférence du ou des concepts liés est précédé d'un point d'interrogation. Tous les concepts d'un ensemble de coréférence ont le même libellé de coréférence.

4.2.2. *Concept maître (DefiningConcept)*

Un concept maître est un concept qui définit l'élément représenté dans un lien de coréférence.

Définition 9 *Un concept maître est le concept source dans un lien de coréférence. Il définit l'élément représenté par tous les concepts de l'ensemble de coréférence.*

La figure 14 présente le graphe de spécification du type DefiningConcept. Un concept maître est un concept dont est issu un lien de coréférence.

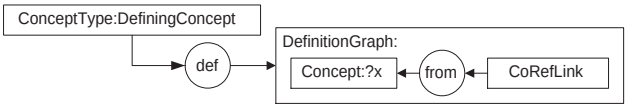

Figure 14. *Graphe de spécification de* DefiningConcept

4.2.3. *Concept lié (BoundConcept)*

Les concepts liés sont les concepts dont le référent est identifié par le concept maître de l'ensemble de coréférence.

Définition 10 *Un concept lié est le concept destination dans un lien de coréférence. Son référent est le même que celui du concept maître de l'ensemble de coréférence.*

La figure 15 présente le graphe de spécification du type BoundConcept. Un concept lié est un concept destination d'un lien de coréférence.

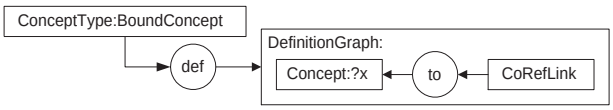

Figure 15. *Graphe de spécification de* BoundConcept

4.3. *Les relations*

4.3.1. *Arc (Arc)*

Les arcs sont les éléments qui permettent de lier les concepts aux relations.

Définition 11 *Un arc lie une relation à un concept dans un graphe conceptuel. Un arc est dit appartenir à la relation et est dit attaché au concept.*

La figure 16 présente le graphe de spécification du type Arc. Les arcs d'un graphe conceptuel sont orientés, et on distingue les arcs orientés vers la relation, les arcs source, de ceux orientés vers le concept, les arcs destination (voir figure 17).

4.3.2. *Relation Conceptuelle (Relation)*

Les relations conceptuelles permettent d'assembler les concepts pour construire une phrase, une idée, une proposition.

Définition 12 *Une relation conceptuelle représente une association entre un ou plusieurs concepts. Une relation conceptuelle est composée d'un libellé de relation qui identifie le type de la relation, et d'arcs qui lient la relation aux concepts associés.*

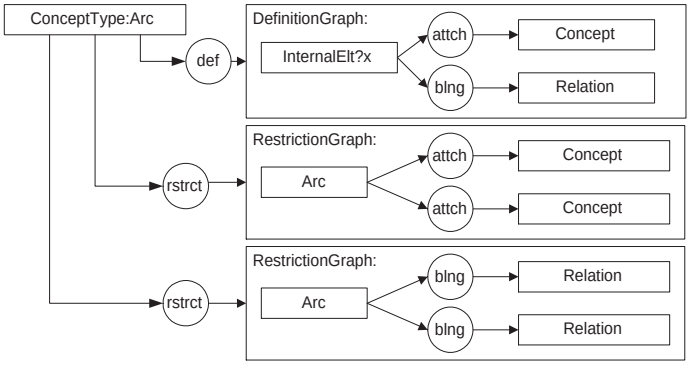

Figure 16. *Graphe de spécification de* Arc

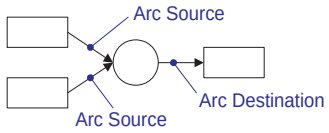

Figure 17. *Source et destination*

La figure 18 présente le graphe de spécification du type Relation. Le graphe de définition montre qu'une relation conceptuelle est associée à un type de relation et est élément d'un graphe conceptuel. Le graphe de restriction précise les cardinalités : une relation ne peut pas être associée à deux types de relation.

Le nombre d'arcs d'une relation conceptuelle est appelé sa *valence*. Parmi les concepts associés, il y a un et un seul concept destination qui est pointé par un arc orienté, et zéro ou plusieurs concepts sources qui sont à l'origine d'un arc orienté (voir figure 17).

Pour une relation de valence n, par convention, les $n-1$ premiers arcs sont orientés vers la relation (arc source) et le $n^{ème}$ est orienté vers un concept (arc destination). Les graphes de règles spécifient cette contrainte. Le premier graphe exprime que s'il existe un arc qui appartient à une relation alors il existe un arc de type destination (possiblement différent) qui appartient à cette relation. Le deuxième graphe exprime que si une relation est liée à un arc destination alors tout autre lien est un arc source.

4.4. *Les types*

4.4.1. *Type (Type)*

Les types sont les éléments qui permettent de définir et d'organiser d'une manière générique la connaissance. Les types sont définis à partir des propriétés communes

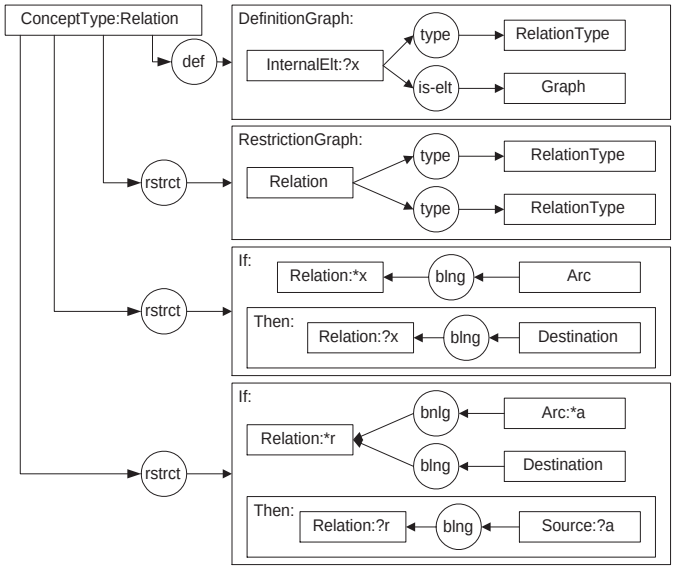

Figure 18. *Graphe de spécification de* Relation

des instances. Les types permettent de regrouper les instances véhiculant une même sémantique. Nous distinguons deux sortes de types : les types de concepts, qui caractérisent les entités, et les types de relation qui caractérisent les liens entre entités.

Définition 13 *Un type représente un ensemble de concepts ou de relations qui ont les mêmes propriétés.*

La figure 19 présente le graphe de spécification de Type. Un type est un élément qui a une relation de sous-typage avec un autre type (relation subt). Un type est défini (relation def) par un graphe de définition qui est nécessairement une projection (relation proj) du graphe de définition du supertype telle que les deux concepts liés (ceux ayant comme référent ?x) soient joints et par un graphe qui montre la syntaxe (relation sntx) pour les éléments du type. Le graphes de restriction montrent qu'un type a un seul graphe de définition et de syntaxe.

4.4.2. *Type de concept (ConceptType)*

Un type de concept est défini, d'une part par son insertion dans la hiérarchie des types de concepts, et d'autre part par un graphe qui montre les relations que ce type de concept doit avoir.

Définition 14 *Un type de concept représente un ensemble de concepts qui ont les mêmes propriétés, c'est-à-dire les mêmes types de relation avec les mêmes types de concept.*

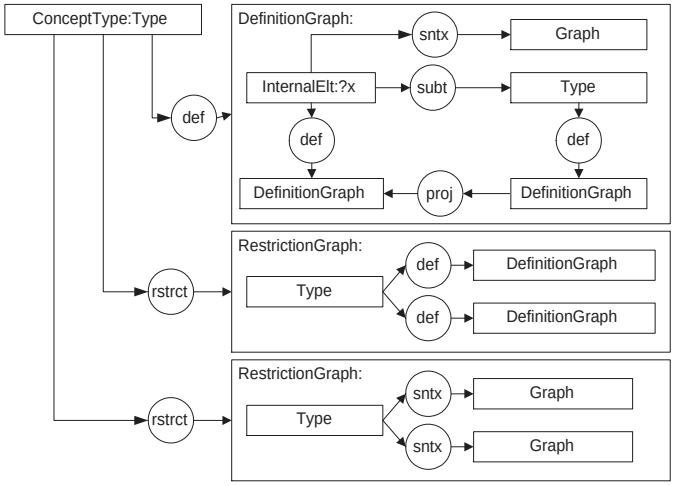

Figure 19. *Graphe de spécification de* Type

La figure 20 présente la définition d'un type de concept. Un type de concept est un type ayant une relation de sous-typage csubt avec un autre type de concept. Les types de concepts sont organisés en une hiérarchie de spécialisation par la relation csubt. À une extrémité de la hiérarchie il y a le type Entity noté $\top_c$, qui représente le type de concept universel dont tous les concepts sont instances, et à l'autre extrémité il y a le type Absurdity noté $\bot_c$ qui représente le type de concept absurde qui n'a aucune instance.

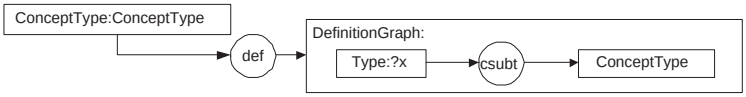

Figure 20. *Graphe de spécification de* ConceptType

Un exemple de définition de type de concept est présenté à la figure 21. Dans la pratique le graphe présentant la syntaxe et la relation de sous-typage peuvent être omis car ils se déduisent du graphe de définition et la définition du type peut être simplifiée comme illustré par la figure 22.

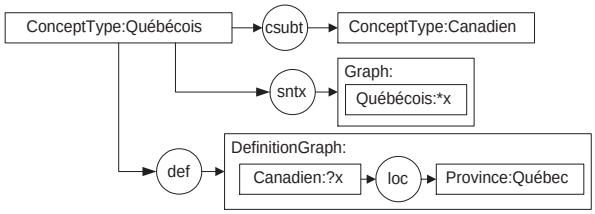

Figure 21. *Graphe de la spécification de* Québécois

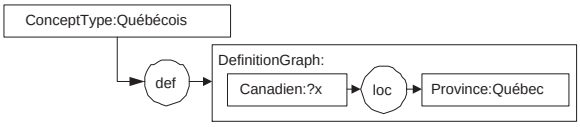

Figure 22. *Graphe simplifié de* Québécois *équivalent à celui de la figure 21*

4.4.3. *Type de relation (RelationType)*

Un type de relation est défini, d'une part par son insertion dans la hiérarchie des types de relation, et d'autre part par un graphe qui montre les types de concepts que ce type de relation peut associer. La figure 23 présente la définition d'un type de relation.

Définition 15 *Un type de relation représente un ensemble de relations qui expriment le même genre d'association entre concepts.*

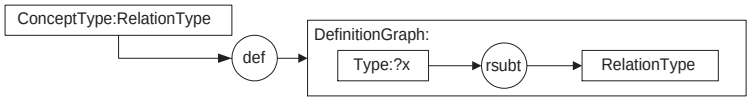

Figure 23. *Graphe de spécification de* RelationType

Les types de relations sont organisés en une hiérarchie de spécialisation par la relation rsubt. À une extrémité de la hiérarchie il y a $\top_r$ qui représente le type de relation universelle dont toutes les relations sont instances et à l'autre extrémité il y a $\bot_r$ qui représente le type de relation absurde qui n'a aucune instance. Dans cette hiérarchie les types de relations sont organisés en fonction de leur valence (Mineau, 2000).

Des exemples de graphes de spécification de type de relation sont illustrés par les figures 24 et 25.

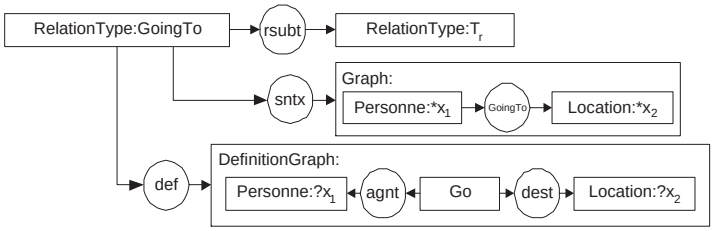

Figure 24. *Graphe de la spécification de la relation* GoingTo

Dans le cas où la relation est binaire, le graphe présentant la syntaxe et la relation de sous-typage peuvent être omis car ils se déduisent du graphe de définition et la définition du type peut être simplifié comme illustré par la figure 26.

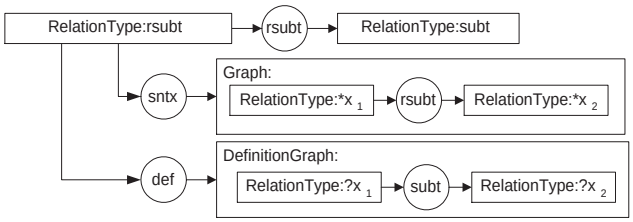

Figure 25. *Graphe de la spécification de la relation* rsubt

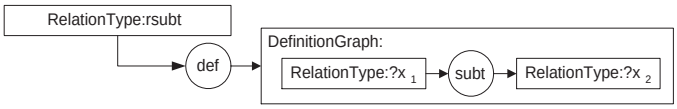

Figure 26. *Graphe simplifié de la spécification de* rsubt

4.5. *Les graphes*

4.5.1. *Graphe de définition (DefinitionGraph)*

Les graphes de définition permettent de spécifier les types de concepts ou de relations.

Définition 16 *Un graphe de définition est un graphe qui définit un type en montrant la forme que doit avoir le graphe impliquant les instances de ce type.*

La figure 27 présente le graphe de spécification de DefinitionGraph. Un graphe de définition est un graphe qui définit un type (relation Def). Un même graphe de définition ne peut pas être associé à deux types différents.

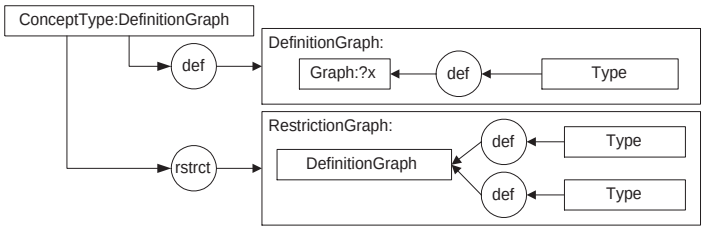

Figure 27. *Graphe de spécification de* DefinitionGraph

4.5.2. *Graphe de restriction (RestrictionGraph)*

Les graphes de restriction permettent dans la spécification d'un type de concept de définir les situations qui ne peuvent pas se produire.

Définition 17 *Un graphe de restriction est un graphe qui spécifie une situation qui n'est pas possible.*

La figure 28 présente le graphe de spécification de RestrictionGraph. Un graphe de restriction est un graphe avec une négation. Le graphe doit être toujours faux et donc sa négation vraie[3].

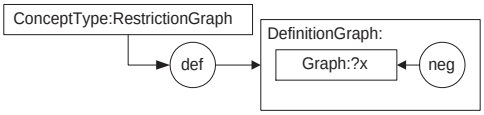

Figure 28. *Graphe de spécification de* RestrictionGraph

4.5.3. *Graphe de règle : Si ... (If)... Alors (Then)*

Les graphes de règle If et Then permettent dans la définition d'un type de concept de spécifier les contraintes que doivent vérifier les instances.

Définition 18 *Un graphe de règles* If *est un graphe de restriction qui spécifie une règle qui doit être toujours vérifiée. Un graphe de règle* Then *est un graphe de restriction qui apparaît dans un graphe de règle* If.

Un graphe de règle est composé d'au moins deux sous-graphes. Le premier représente la prémisse, c'est-à-dire, la clause If. Le deuxième représente la conclusion ; il est réduit à un seul concept de type Then. Si le graphe ou les graphes définis dans la clause If sont vrais alors le graphe de la clause Then doit être également vrai. La figure 29 présente les graphes de spécification de If et Then.

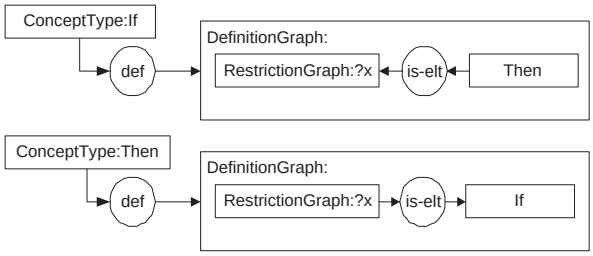

Figure 29. *Graphe de spécification de* If *et* Then

3. Nous rappelons que l'opération de projection est injective. Si deux concepts sont distincts alors ils se projettent sur des concepts distincts

4.6. *Contexte*

4.6.1. *Contexte (Context)*

Les contextes sont indispensables pour segmenter l'information. Ils définissent des espaces de validité pour les assertions. Les contextes ont été utilisés dans différents domaines par la communauté travaillant avec les graphes contextuels. Les contextes sont importants pour la définition de la connaissance (Esch, 1993; Esch, 1994a; Esch, 1994b), le traitement de la langue naturelle (Dick, 1994; Moulin, 1993), la représentation des relations temporelles (Moulin, 1993), dans les systèmes multi-agents (Moulin, 1995) et les systèmes objets (Sowa, 1996).

Bien que les contextes aient été souvent utilisés, il n'y a pas de consensus sur leur utilisation. Dans le standard en cours d'élaboration (ISO, 2005), un contexte est défini comme un concept dont le référent est un graphe conceptuel. Nous avons étendu cette définition en distinguant ce graphe conceptuel du graphe contextuel qui détermine sa validité. Nous développons ci-dessous cette définition.

Un graphe g représentant une proposition vraie est appelé une assertion. Excepté pour les vérités universelles, la valeur de vérité d'une proposition dépend d'autres assertions. Ces assertions définissent les conditions selon lesquelles la proposition est vraie. Ces assertions spécifient les situations où le graphe g est une assertion. L'ensemble de toutes les situations où le graphe g est une assertion est appelé sa *portée*. Une assertion ne peut être établie sans préciser sa portée. Les graphes sont donc toujours définis en relation à une situation. Par défaut, dans un système basé sur les graphes conceptuels, il existe une feuille d'assertion, dite *feuille d'assertion universelle*. Toutes les vérités universelles sont faites sur la feuille d'assertion universelle. Comme toutes les assertions sont établies en relation avec une situation (qui peut être la feuille d'assertion universelle), il est nécessaire de lier d'une manière formelle une proposition à l'ensemble des conditions selon lesquelles cette proposition est une assertion. Cette paire <condition,assertion> est appelée un *contexte*.

Un contexte est une feuille d'assertion dont l'existence dépend de graphes conceptuels qui décrivent les conditions de son existence. Tous les graphes du contexte dépendent des mêmes conditions. L'assertion « Marie pense que Pierre l'aime » est universellement vraie. Mais est-ce que Pierre aime réellement Marie ? Nous ne pouvons rien affirmer à ce sujet, excepté que « Pierre aime Marie » dans les pensées de Marie. Les pensées de Marie forment un contexte dans lequel la proposition « Pierre aime Marie » est vraie. La figure 30 montre l'assertion ; Les figures 31 et 32 montrent respectivement l'*intention* et l'*extension* du contexte associé. Les graphes d'intention et d'extension peuvent être dérivés automatiquement à partir de l'assertion de la figure 30.

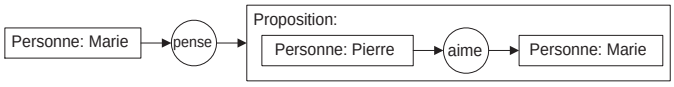

Figure 30. *La proposition « Marie pense que Pierre l'aime »*

Figure 31. *L'extension du contexte des pensées de Marie*

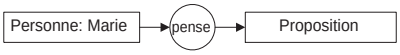

Figure 32. *L'intention du contexte des pensées de Marie*

Un autre exemple présenté dans la figure 33 est la proposition suivante : « Marie pense que Pierre est beau ».

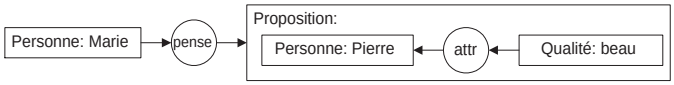

Figure 33. *La proposition « Marie pense que Pierre est beau »*

D'une manière formelle un contexte C est spécifié par deux ensembles de graphes T et G. T définit les conditions sous lesquelles C existe, et G est l'ensemble des graphes vrais dans ce contexte.

Définition 19 *Un contexte est composé de deux parties : la première partie, l'intention, est un ensemble de graphes qui décrit les conditions sous lesquelles les graphes de la deuxième partie, l'extension, sont vrais.*

La figure 34 présente le graphe de spécification de Context. Un contexte est un élément interne associé par une relation d'intention (intn) à un ou plusieurs graphes et par une relation d'extension (extn) à un ou plusieurs autres graphes.

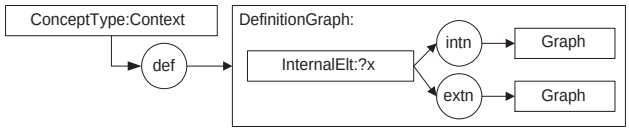

Figure 34. *Graphe de spécification de* Context

Le lecteur intéressé par la notion de contexte trouvera plus de détails sur la mécanique des contextes et la définition de contexte formel ainsi que des opérateurs associés dans (Mineau *et al.*, 1997).

4.7. *Les relations conceptuelles du langage*

Les relations conceptuelles du langage utilisées pour décrire le formalisme même des graphes conceptuels permettent de structurer les concepts en graphes conceptuels.

Nous les avons introduites dans les différents graphes de spécification des types de concepts. Le lecteur intéressé trouvera en annexe la liste exhaustive avec leur définition.

5. Conclusion

Nous avons proposé dans cet article un métamodèle pour le formalisme des graphes conceptuels. La spécification des éléments du métamodèle a été faite en utilisant les graphes conceptuels eux-mêmes montrant ainsi la puissance d'expression de ce formalisme. Les graphes conceptuels peuvent être utilisés pour représenter les différents niveaux de modélisation de l'univers du discours.

Les applications des techniques de métamodélisation sont nombreuses et nous travaillons à l'utilisation de ces techniques au domaine de la gestion de la qualité de service dans la livraison de ressources multimédias (Gerbé *et al.*, 2003a; Kerherve *et al.*, 2006). Il s'agit, dans ce contexte, de manipuler, transformer et instancier des modèles de qualité de service. Ces modèles décrivent les caractéristiques des ressources multimédias elles-même, les caractéristiques des systèmes de stockage (serveurs) et de distribution (réseaux) et les préférences des usagers. Le problème est l'adaptation de tous les maillons de la chaîne de distribution aux exigences exprimées par l'usager. Nous pensons que les graphes conceptuels avec leurs mécanismes de généralisation et de dérivation sont particulièrement bien adaptés à cette tâche.

Comme nous l'avons déjà indiqué, l'utilisation des graphes conceptuels dans le cadre du Web sémantique,pourrait être un atout important pour l'interopérabilité des systèmes. Nous pensons que les graphes conceptuels de par leur puissance et leur expressivité proche de la langue naturelle serait un bon langage pivot dans ce domaine (Gerbé *et al.*, 2002) et nous travaillons actuellement à la métamodélisation de règles de transformation qui permettraient de passer d'un formalisme de réprésentation aux graphes conceptuels et récproquement.

6. Bibliographie

Baget J.-F., Mugnier M.-L., « Extensions of Simple Conceptual Graphs : The Complexity of Rules and Constraints », *Journal of Artificial Intelligence Research*, vol. 16, p. 425–465, 2002.

Berners-Lee T., « Conceptual Graphs and the Semantic Web », February, 2001. available at http ://www.w3.org/DesignIssues/CG.html.

Bernstein P., Halevy A., R. P., « A vision for management of complex models », *SIGMOD Rec.*, vol. 29, n° 4, p. 55–63, 2000.

Chein M., Mugnier M.-L., « Conceptual Graphs : Fundamental Notions », *Revue d'intelligence artificielle*, vol. 6, n° 4, p. 365–406, 1992.

Chein M., Mugnier M.-L., « Types and Coreference in Simple Conceptual Graphs », *Proceedings of the International Conference on Conceptual Structures, ICCS2004*, n° 3127 in *LNAI*, Springer, Hunstville, USA, p. 303–318, 2004.

Corby O., Dieng R., Hebert C., « A Conceptual Graph Model for W3C Resource Description Framework », *Proceedings of the International Conference on Conceptual Structures, ICCS2000*, p. 468–482, 2000.

Delteil A., Dieng R., Faron-Zucker C., « Extension of RDFS Based on the CGs Formalism », *in* H. Delugach, G. Stumme (eds), *Proceedings of the 9th International Conference on Conceptual Structures, ICCS 2001*, n° 2120 in *LNAI*, Springer-Verlag, Stanford, CA, USA, p. 275–289, July/August, 2001.

Dick J., « Using Contexts to Represent Text », *in* J. Dick, J. Sowa, W. Tepfenhart (eds), *Proceedings of the Second International Conference on Conceptual Structures, ICCS1994*, n° 835 in *LNAI*, Springer-Verlag, College Park, Maryland, USA, p. 196–213, August, 1994.

Esch J., « Contexts as White Box Concepts », *in* G. Mineau, B. Moulin, J. Sowa (eds), *Proceedings of the First International Conference on Conceptual Structures, ICCS1993*, n° 699 in *LNAI*, Springer-Verlag, Quebec City, Quebec, Canada, p. 17–29, August, 1993.

Esch J., « Contexts and Concepts, Abstraction Duals », *in* J. Dick, J. Sowa, W. Tepfenhart (eds), *Proceedings of the Second International Conference on Conceptual Structures, ICCS1994*, n° 835 in *LNAI*, Springer-Verlag, College Park, Maryland, USA, p. 175–184, August, 1994a.

Esch J., « Contexts, Canons and Coreferent Types », *in* J. Dick, J. Sowa, W. Tepfenhart (eds), *Proceedings of the Second International Conference on Conceptual Structures, ICCS1994*, n° 835 in *LNAI*, Springer-Verlag, College Park, Maryland, USA, p. 185–195, August, 1994b.

Favre J.-M., Estublier J., M. B., *L'ingénierie dirigée par les modèles : au-delà de MDA*, Édition Hermès-Lavoisier, Février, 2006.

Flatscher R., « Metamodeling in EIA/CDIF—meta-metamodel and metamodels », *ACM Trans. Model. Comput. Simul.*, vol. 12, n° 4, p. 322–342, 2002.

Gerbé O., Kerhervé B., Srinivasan U., « Model Operations for Quality-Driven Multimedia Delivery », *in* B. Ganter, A. de Moor (eds), *Using Conceptual Structure : Contributions to ICCS2003*, Shaker-Verlag, Dresden, Germany, p. 83–96, August, 2003a.

Gerbé O., Mineau G., « The Conceptual Graph Formalism as an Ontolingua for Web-Oriented Representation Languages : The RDF Schema Case Study », *in* U. Priss, D. Corbett, G. Angelova (eds), *Proceedings of the 10th International Conference on Conceptual Structures, ICCS 2002*, Springer-Verlag, Borovets, Bulgaria, p. 205–219, July, 2002.

Gerbé O., Mineau G., Keller R., « Conceptual graphs and Metamodeling », *in* H. Delugach, G. Stumme (eds), *Proceedings of the 9th International Conference on Conceptual Structures, ICCS 2001*, Springer-Verlag, Stanford, CA, USA, p. 245–259, July/August, 2003b.

Groupe Ingénierie des modèles sous la driection de Jean Bézivin, Ingénierie des modèles - logiciels et systèmes, ARAGO n° 30, Observatoire Français des Techniques Avancées, 2004.

ISO/JTC1/SC 32, *Information technology - Common Logic (CL) - A Framework for a Family of Logic-Based Languages*. December, 2005, Final Committee Draft ISO/IEC.

Kerherve B., NGuyen K., Gerbé O., B. J., « A framework for Quality-Driven Delivery in Distibuted Multimedia Systems », *Proceedings of International Conference on Internet and Web Applications and Service)*, Guadeloupe, France, 2006.

Mineau G., « The Extensional Semantics of the Conceptual Graph Formalism », *in* B. Ganter, G. Mineau (eds), *Proceedings of the 8th International Conference on Conceptual Structures, ICCS2000*, n° 1867 in *LNAI*, Springer-Verlag, Darmstadt, Germany, p. 221–234, August, 2000.

Mineau G., Gerbé O., « Contexts : a formal definition of worlds of assertions », *in* H. Delugach, M. Keeler, L. Searle, J. Sowa (eds), *Proceedings of the 5th International Conference on Conceptual Structures, ICCS1997*, n° 1257 in *LNAI*, Springer-Verlag, Seattle, Washington, USA, p. 80–94, August, 1997.

Moulin B., « The Representation of Linguistic Information in an Approach Used for Modeling Temporal Knowledge in Discourses », *in* G. Mineau, B. Moulin, J. Sowa (eds), *Proceedings of the First International Conference on Conceptual Structures, ICCS1993*, n° 699 in *LNAI*, Springer-Verlag, Quebec City, Quebec, Canada, p. 182-204, August, 1993.

Moulin B., « Discourse Spaces : A Pragmatic Interpretation of Contexts », *in* G. Ellis, R. Levinson, W. Rich, J. Sowa (eds), *Proceedings of the Third International Conference on Conceptual Structures, ICCS1995*, n° 954 in *LNAI*, Springer-Verlag, Santa Cruz, CA, USA, p. 89–104, August, 1995.

Mugnier M.-L., « On Generalization/Specialization for Conceptual Graphs », *Journal of Experimental and Theorical Artificial Intelligence*, vol. 7, n° 3, p. 325–344, 1995.

Mugnier M.-L., Chein M., « Représenter des connaissances et raisonner avec des graphes », *Revue d'intelligence Artificielle*, vol. 10, n° 1, p. 7–56, 1996.

Object Management Group, *Meta Object Facility (MOF) 2.0 Core Specification*. 2006, OMG Adopted Specification, formal/06-01-01.

Sowa J., *Conceptual Structures : Information Processing in Mind and Machine*, Addison-Wesley, 1984.

Sowa J., « Processes and Participants », *in* P. Eklund, G. Ellis, G. Mann (eds), *Proceedings of Fourth International Conference on Conceptual Structures, ICCS1996*, LNAI, Springer-Verlag, Sydney, Australia, p. 1–22, August, 1996.

Sowa J., *Knowledge Representation : Logical, Philosophical, and Computational Foundations*, BrooksCole, 2000.

Wermelinger M., « Conceptual Graphs and First-Order Logic », *in* G. Ellis, R. Levinson, W. Rich, J. Sowa (eds), *Proceedings of the Third International Conference on Conceptual Structures, ICCS1995*, vol. 954 of *LNAI*, Springer-Verlag, Santa Cruz, CA, USA, p. 323–337, August, 1995.

Annexe A

Les relations conceptuelles du langage

Cette annexe détaille chacune des relations conceptuelles du langage présentées à la figure 6 (page 265). Les relations conceptuelles du langage permettent de structurer les concepts en graphes conceptuels. Ci-dessous nous donnons par ordre alphabétique la définition de chacune des relations conceptuelles du langage. La figure 35 (page 286) regroupe les graphes de spécifications des relations.

attaché à / attached to (attch)

La relation attch associe un arc au concept auquel il est attaché. Voir figure 35-1.

appartient à / belongs to (blng)

La relation blng est une relation d'appartenance. Elle associe un arc à la relation à laquelle il appartient. Voir figure 35-2.

conforme / conforms (conf)

La relation de conformité conf associe un référent aux types auxquels il se conforme. Un référent $x1$ est conforme à un type $x2$ si et seulement si il existe un concept dont $x1$ est le référent et qui a une relation type avec $x2$. Voir figure 35-3.

sous-type de concept / concept subtype (csubt)

La relation csubt est une relation de sous-typage et une spécialisation de la relation subt. Elle associe des types de concepts. Voir figure 35-4.

définit / defines (def)

La relation def associe un graphe de définition au type de concept qu'il spécifie. Voir figure 35-5.

extension (extn)

La relation extn associe un contexte à un graphe d'extension. Voir figure 35-6.

de / from (from)

La relation from associe un lien de coréférence au concept maître qui est la source de la coréférence. Voir figure 35-7.

intension (intn)

La relation intn associe un contexte à un graphe d'intension. Voir figure 35-8.

est élément de / is element of (is-elt)

La relation is-elt associe un élément interne : concept, relation ou arc au graphe dont il est un élément. Voir figure 35-9.

projection (proj)

La relation proj est une relation de dérivation. Elle associe un graphe à un graphe dérivé. Voir figure 35-10.

pointe vers /points (pnt)

La relation pnt est une relation d'orientation. Elle associe un arc orienté à l'élément vers lequel l'arc pointe. Voir figure 35-11.

réfère à / refers to (ref)

La relation ref associe un concept à son référent. Voir figure 35-12.

représente / represents (rep)

La relation rep associe un référent à l'élément qu'il représente. Le référent est un représentant de l'élément dans le système. Voir figure 35-13.

restriction (rstrct)

La relation rstrct associe un type aux graphes de restriction qui contraignent les instances du type. Voir figure 35-14.

sous-type de relation / relation subtype (rsubt)

La relation rsubt est une relation de sous-typage et une spécialisation de la relation subt. Elle associe des types de relations. Voir figure 35-15.

syntaxe / syntax (sntx)

La relation sntx associe un type à son graphe de syntaxe qui montre comment écrire ou représenter les instances du type. Voir figure 35-16.

sous-type / subtype (subt)

La relation subt est une relation de sous-typage. Elle associe deux types, le premier étant sous-type du deuxième. Le type est plus général et peut être mis en lieu et place du sous-type. Voir figure 35-17.

type

La relation type associe un concept au type dont il est une instance. La figure 35-18 présente la définition de la relation type.

1 $[RelationType : attch] \rightarrow$
$(def) \rightarrow [DefinitionGraph : [Arc :?x_1] \rightarrow (T_r) \rightarrow [Concept :?x_2]]$
2 $[RelationType : blng] \rightarrow$
$(def) \rightarrow [DefinitionGraph : [Arc :?x_1] \rightarrow (T_r) \rightarrow [Relation :?x_2]]$
3 $[RelationType : conf] \rightarrow$
$(subt) \rightarrow [RelationType : T_r]$
$(sntx) \rightarrow [Referent : *x_1] \rightarrow (conf) \rightarrow [ConceptType : *x_2]]$
$(def) \rightarrow [DefinitionGraph :$
$[Referent :?x_1] \leftarrow (ref)[Concept] \leftarrow (type) \leftarrow [ConceptType :?x_2]]$
4 $[RelationType : csubt] \rightarrow$
$(def) \rightarrow [DefinitionGraph : [ConceptType :?x_1] \rightarrow (T_r) \rightarrow [ConceptType :?x_2]]$
5 $[RelationType : def] \rightarrow$
$(def) \rightarrow [DefinitionGraph : [DefinitionGraph :?x_1] \rightarrow (T_r) \rightarrow [Type :?x_2]]$
6 $[RelationType : extn] \rightarrow$
$(def) \rightarrow [DefinitionGraph : [Context :?x_1] \rightarrow (T_r) \rightarrow [Graph :?x_2]]$
7 $[RelationType : from] \rightarrow$
$(def) \rightarrow [DefinitionGraph : [CoRefLink :?x_1] \rightarrow (T_r) \rightarrow [DefiningConcept :?x_2]]$
8 $[RelationType : intn] \rightarrow$
$(def) \rightarrow [DefinitionGraph : [Context :?x_1] \rightarrow (T_r) \rightarrow [Graph :?x_2]]$
9 $[RelationType : is - elt] \rightarrow$
$(def) \rightarrow [DefinitionGraph : [Graph\_Elt :?x_1] \rightarrow (T_r) \rightarrow [Graph :?x_2]]$
10 $[RelationType : proj] \rightarrow$
$(def) \rightarrow [DefinitionGraph : [Graph :?x_1] \rightarrow (T_r) \rightarrow [Type :?x_2]]$
11 $[RelationType : pnt] \rightarrow$
$(def) \rightarrow [DefinitionGraph : [Arc :?x_1] \rightarrow (T_r) \rightarrow [Graph_Elt :?x_2]]$
12 $[RelationType : ref] \rightarrow$
$(def) \rightarrow [DefinitionGraph : [Concept :?x_1] \rightarrow (T_r) \rightarrow [Referent :?x_2]]$
13 $[RelationType : rep] \rightarrow$
$(def) \rightarrow [DefinitionGraph : [Referent :?x_1] \rightarrow (T_r) \rightarrow [Element :?x_2]]$
14 $[RelationType : rstrct] \rightarrow$
$(def) \rightarrow [DefinitionGraph : [Type :?x_1] \rightarrow (T_r) \rightarrow [RestrictionGraph :?x_2]]$
15 $[RelationType : rsubt] \rightarrow$
$(def) \rightarrow [DefinitionGraph : [RelationType :?x_1] \rightarrow (subt) \rightarrow [RelationType :?x_2]]$
16 $[RelationType : sntx] \rightarrow$
$(def) \rightarrow [DefinitionGraph : [Type :?x_1] \rightarrow (T_r) \rightarrow [Graph :?x_2]]$
17 $[RelationType : subt] \rightarrow$
$(def) \rightarrow [DefinitionGraph : [Type :?x_1] \rightarrow (T_r) \rightarrow [Type :?x_2]]$
18 $[RelationType : type] \rightarrow$
$(def) \rightarrow [DefinitionGraph : [Concept :?x_1] \rightarrow (T_r) \rightarrow [ConceptType :?x_2]]$

Figure 35. *Graphes de spécifications des relations conceptuelles.*