Documentation de l'ontologie de Prolexbase

Mouhamad Ndiankho THIAM, Joshua Amavi

24 Avril 2015

Résumé

Prolexbase est une base de données de noms propres conçue pour être exploitée dans plusieurs contextes. Elle permet des représenter des noms propres, de tisser des liens entre eux, de donner des informations spécifiques par rapport à une langue données. Elle est accessible sur internet [1] et son format actuel est en SQL. Ce serait intéressant de pouvoir l'exploiter à partir d'autres formats, ce qui étendrait beaucoup son utilisation. C'est ce qui motive son intégration comme ressource dans le Web Sémantique.

La portée de ce travail Le but de ce document est de présenter l'ontologie de Prolexbase. Cette ontologie est déduite de manière logique de la docuementation, mais aussi à partir de contenus de la base. Dans ce qui suit, nous allons expliquer un à un les différents concepts et relations de cette ontologie en les mettant en rapport autant que possible avec leurs correspondants dans la conception de la base de données. Nous allons procéder par niveaux.

1 La partie métaconceptuelle

Tel que précisé par les concepteurs, cette partie est indépendante de la langue. On y retrouve trois différents concepts ainsi que trois relations. Il faut seulement noter que bien que la notion de pivot est inhérente au niveau conceptuel, nous la présentons ici pour surtout faire surgir ses relations avec des éléments de ce niveau.

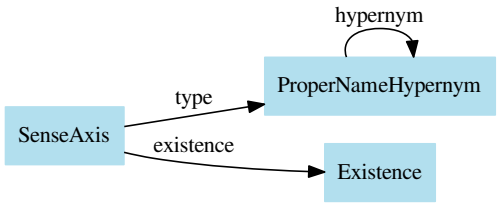

FIGURE 1 – Les classes et leurs relations

1. www.cnrtl.fr/lexiques/prolex/

Dans la *figure 1* on a les différents concepts et relations utilisées dans cette couche. Ils sont à mettre en rapport directement avec la *figure 2*.

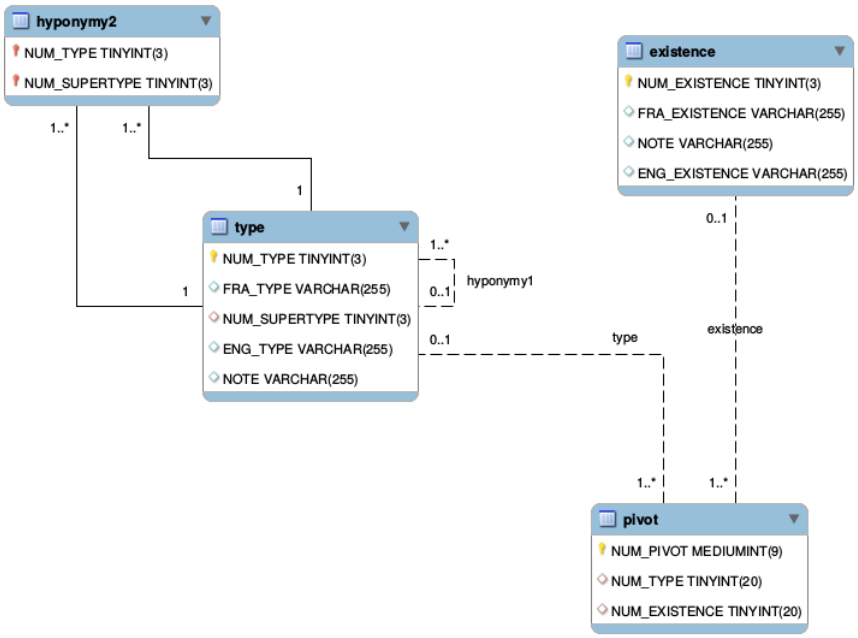

FIGURE 2 – Diagramme Entité-Association

Regardons d'abord les concepts.

1.1 Les concepts

Dans la *figure 1* on se retrouve avec les concepts de `SenseAxis` (`Pivot`), `ProperNameHypernym` et `Existence`. Chacun d'entre eux représente une classe ou entité et est utilisé à travers des instances. C'est les mêmes entités que l'on retrouve dans la *figure 2* où la table *type* correspond à `ProperNameHypernym`, sauf pour *hyponym2* qui est en fait une relation matérialisée par une entité.

1. **SenseAxis :** Il caractérise une abstraction du nom propre indépendamment de la langue, de ses variantes et dérivations. Chaque pivot est relié à son type et à son existance. Dans cette couche métaconceptuelle on ne voit pas de relations entre pivots mais seulement entre un pivot et ces deux types d'informations. Un pivot n'a qu'une seule existance et un seul type.
2. **Existence :** Cette entité représente la façon par laquelle existe un nom propre. Il y a ainsi trois types d'existences :
   — historique (`historical`)
   — religieux (`religious`)
   — fictive (`fictitious`)
   Pour le matérialiser, nous avons défini trois individus dans l'ontologie ; chacun représentant une de ces existences. C'est ce qu'on observe dans la *figure 3*.

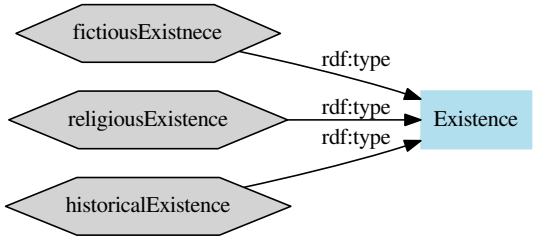

FIGURE 3 – L'existence

3. **ProperNameHypernym :** Cette entité regroupe à la fois les types et les supertypes qui permettent de faire une classification homogène des noms propres. Les supertypes donnent une information minimaliste sur la sémantique. On se retrouve avec quatre supertypes :

   anthroponymes : noms d'êtres humains, animaux, rebots, ... ;

   toponymes : noms de lieux ;

   ergonymes : noms désignant ce qui concret ou abstrait produit par l'homme ;

   pragmonymes : noms d'évènements ;

   Au delà de ces supertypes, on retrouve trente types. chaque supertype peut être divisé en plusieurs types. Nous reviendrons sur ces relations hiérarchiques dans la partie réservée à l'hypéronymie. La *table* 1 (The Prolex typology) tirée de [2] est beaucoup plus claire.

Proper Name						
Anthroponym			Toponym		Ergonym	Pragmonym
Individual	Collective					
		Group		Territory		
Celebrity First Name Patronymic Pseudo-anthroponym	Dynasty Ethnonym	Association Ensemble Firm Institution Organization	Astronym Building City Geonym Hydronym Way	Country Region Supra-national	Object Product Thought Vessel Work	Disaster Event Feast History Meteorology

TABLE 1 – Typologie de Prolex

Remarque : Nous avons déclaré `ProperNameHypernym` comme sous-classe de `skos:Concept`. Ainsi tous les individus que nous utilisons pour représenter les types et les supertypes sont aussi des concepts du point de vue de **SKOS**.

1.2 Les relations

Une fois que l'on connait les entités qui entrent en jeu, place maintenant aux relations qui peuvent les lier. Les deux premières partent du pivot alors que la dernière est réflexive sur l'entité `ProperNameHypernym`.

1. **type :** Comme on l'a dit plus haut cette relation permet de désigner le type d'un pivot. On peut par exemple dire que le pivot représentant Paris est de type ville (`City`). Cette relation est fonctionelle c'est-à-dire qu'un pivot est relié à un et un seul type.
2. **existence :** Elle a les mêmes caractéristiques que type (fonctionelle). Seulement sa patte est reliée à un élément `Existence` pour préciser ainsi la raison d'existence du nom propre.
3. **hypéronymie :** Elle est beucoup plus complexe que les deux précédentes. Elle relie deux instances de `ProperNameHypernym` et permet ainsi de faire ressortir la relation hiérarchique de ces éléments. Elle se subdivise en deux relations :
   — **hypéronymie principale** : Elle correspond à la subsumption (*is-a*). Par exemple une région est toujours un territoire qui est toujours un toponyme. La *figure 4* énumère tous les individus de la classe `ProperNameHypernym` et les dispose de manière hiérarchique, telle que présentée dans la *table 1* avec la relation *pH* (`primaryHypernym`). Cette relation est fonctionelle.

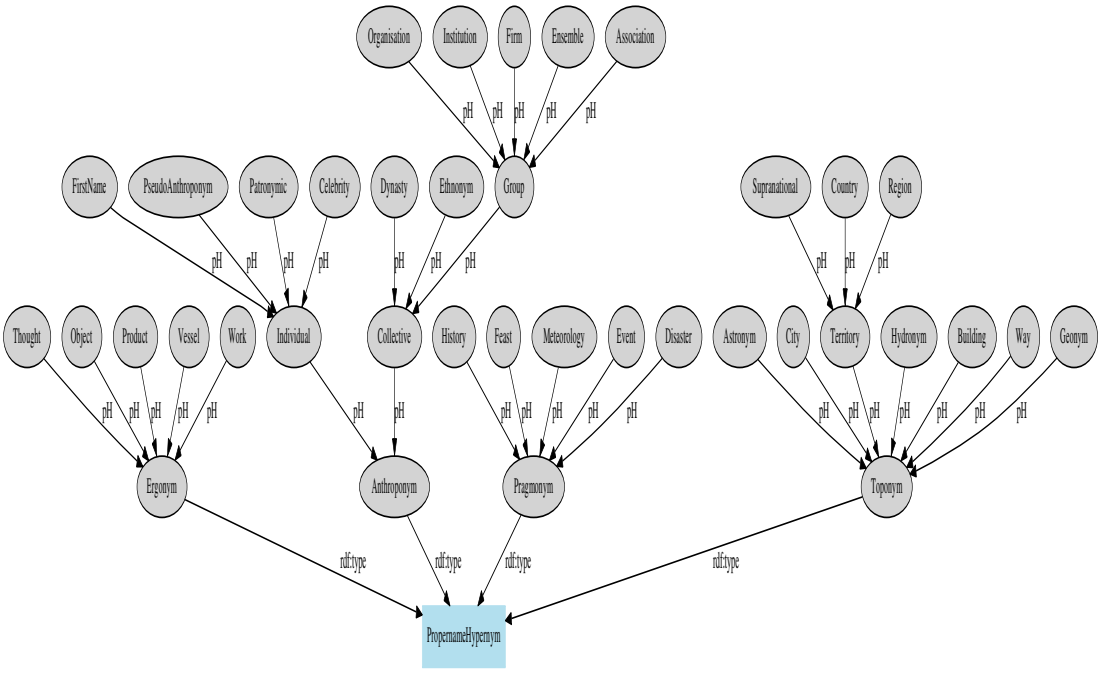

FIGURE 4 – L'hypéronymie primaire

— **hypéronymie secondaire :** Sa différence avec la principale est qu'elle n'est pas toujours utilisée. Par exemple le nom d'une ville peut être vu comme un `Collective Anthroponym`. Elle est plus proche de la notion *peut-être* (`canBe`) et est différente de celle de *est* (`is-a`). La *figure* 5 reproduit à partir d'individus la seconde hypéronymie telle que présentée dans la *table* 2. Cette relation n'est pas fonctionelle, donc un type peut être lié par hypéronymie secondaire à plusieurs supertypes.

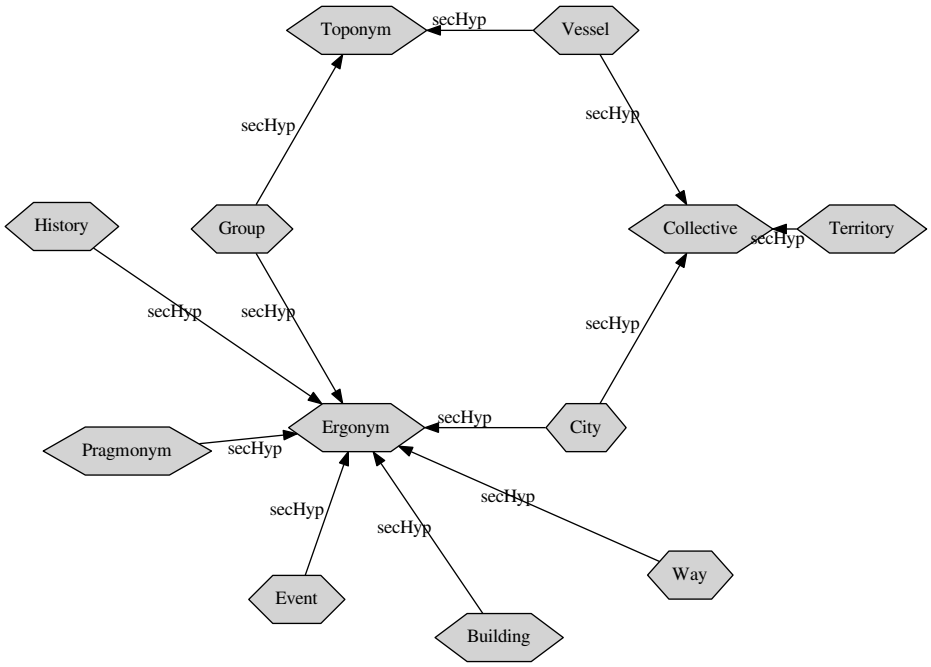

FIGURE 5 – L'hypéronymie secondaire

Types	*Secondary hypernym*
Territory	Collective anthroponym
City	Collective anthroponym Ergonym
Building Way	Ergonym
Event Feast History	
Group	Ergonym Toponym
Vessel	Collective anthroponym Toponym

TABLE 2 – La seconde Typologie de Prolex

Remarque : En dessous des relations d'hypéronymie nous avons utilisé **SKOS** en déclarant *primaryHypernym* comme équivalente à *skos :broaderTransitive* et *secondaryHypernym* à *skos :narrowerTransitive*.

Remarques : Pour permettre des inférences, nous avons défini des relations inverses à savoir `isTypeOf`, `isExistenceOf`, et `hyponym`.

1.3 Exemple :

Pour clore cette partie, nous illustrons ce que nous avons présenté à travers un exemple (*figure* 6).
Dans cet exemple on représente la partie méta-conceptuelle de Guadeloupe. Le pivot le représentant (`:Guadeloupe`) est rattachée à une existance historique (`:HistoricalExistence`). Il est aussi relié à son type région (`:Region`). Ce dernier a une relation d'hypéronymie primaire avec territoire (`:Territory`) qui a, à son tour cette même relation avec toponyme (`:Toponym`). On a aussi que territoire a une relation d'hypéronymie secondaire avec collective (`:Collective`) qui a, à son tour une relation d'hypéronymie principale avec anthroponyme (`:Anthroponym`).

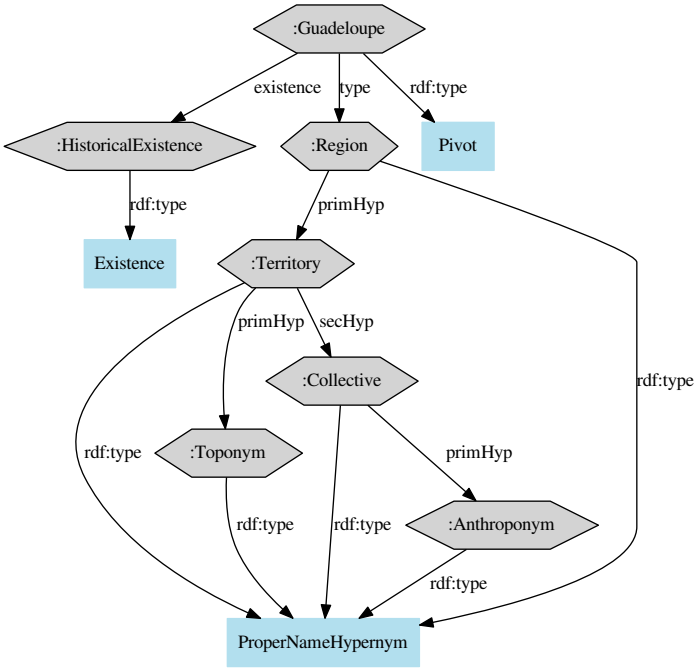

FIGURE 6 – Exemple pour la couche métaconceptuelle

2 La partie conceptuelle

Dans cette partie le concept central utilisé est celui de pivot (`SenseAxis`). Nous l'avons déja présenté dans la première partie. Ici on met surtout l'accent sur les relations que les pivots peuvent entretenir. C'est ce qu'illustre la *figure* 7. On y voit deux pivots qui sont liés par une certaine relation. On a pas précisé le type de relation car le shéma ne change pas quelque soit cette relation. Nous allons alors les détailler dans la suite. Il y a aussi le concept de `SynSet` qui regroupe l'ensemble de pivots reliés par une synonymie (dans le sens de **WordNet**). La *figure* 8 donnent la manière dont fonctionne le SynSet.

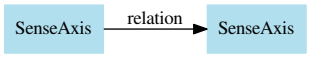

FIGURE 7 – La couche conceptuelle

2.1 Les relations mises en jeu

Ces relations sont très importante dans Prolexbase car elle ne fournissent pas d'informations sur un pivot mais lie plutot des pivots. On peut par exemple

dire qu'un nom propre est le quasi-synonyme d'un autre, ou encore est inclus dans cet autre ou bien qu'il est accessible à partir de lui.

2.1.1 la quasi-synonymie

Elle permet de dire que deux pivots désignent deux points de vue différents du même référent. Ainsi on peut dire que le vrai nom et le pseudonyme d'un artiste désignent la même personne (quasi-synonymes), mais chacun révèle une facette différente. Pour cela nous avons défini la relation *quasiSynonym* qui,à son tour, a trois sous-relaions qui précisent chacune sur quel motif est basé ce point de vue. Ce motif est appelé le diasystème. Voici les trois diasystèmes :

- **diachronique** : Ceci met en exergue une variation du nom selon le temps. Par exemple on peut dire que le `Burkina Faso` s'appelait auparavant `La Haute Volta`. La sous-relation est nommée *diachronicSynonym*.
- **diaphasique** : Elle désigne une variante liée à l'usage. Par exemple dans le milieux touristique on désignera `Paris` par `La Ville Lumière` ou le `Sénégal` par `Le Pays de la Téranga`. Nous la représentons par la relation *diaphasicSynonym*.
- **diastratique** : Cette variante quant à elle révèle la différence socio-culturelle. Par exemple le chanteur-compositeur américain `Bob Dylan` est très célèbre dans le milieu artistique alors que son vrai nom `Robert Allen Zimmerman` est peu connu. Nous la représentons par la relation *diastraticSynonym*.

Remarques : Il faut remarquer que la quasi-synonymie ainsi que ses trois sous-relations sont toutes transitives et symétriques.

2.1.2 Les goupes de synonymes

C'est ici que l'on traite les SynSets. pour dire qu'un pivot appartient à un SynSet on utilise la relation *member* qui va vers ce pivot. En dehors de pouvoir désigner les membres on a la possibilité de dire que parmi tous ces membres il y en a un qui est canonique (c'est à dire qui est utilisé de manière privilégiée). Ceci se fait gràce à la relation *canonicPivot*. Elle est fonctionelle car on ne peut avoir qu'un seul canonique dans un même SynSet. Pour faciliter nous avons mis cette dernière comme sous-relation de *member*. Anisi dire qu'un pivot est canonique suffit pour dire qu'il fait partie. La *figure 8* qui suit nous illustre cela :

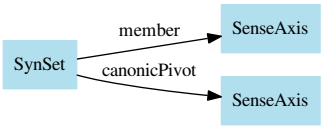

FIGURE 8 – Le SynSet

2.1.3 la méronymie

Cette raltion traduit l'inclusion au sens mathématique. Au départ elle permattait de reproduire les imbrications des toponymes. Mais elle est élargie aux autres. Il sera aisé de dire que `Paris` est dans la `France` qui est aussi dans l'`Europe`. Pour cela nous avons défini la relation par *hasMeronym* et son inverse par *holonym*.

2.1.4 l'accessibilité

Cette relation est très forte pour la définition d'un nom propre à partir d'un autre déja connu. On peut par exemple expliquer `Berlin` à partir de l' `Allemagne` (supposé déja connu) en disant que c'est sa capitale. Nous l'avons spécialisée en plusieurs pour préciser le sujet de la relation. La relation mère est désignée par *accessible*.

- **parent** : Ici on précise que nos deux pivots entretiennent un lien de parenté. Elle se traduit par la relation *relativeAccessible*.
- **capitale** : Dans ce cas il s'agit de dire que le premier est la capitale du second comme dans l'exemple précédent. Nous avons nommé la relaion *capitalAccessible*.
- **dirigeant politique** : Un dirigeant politique pourra être lié à ce qu'il dirige. On la désigne par *leaderAccessible*.
- **fondateur** : On l'utilise pour relier un fondateur à sa fondation, et on l'appele *founderAccessible*.
- **élève** : Celle-ci permet de mettre en relation un élève ou disciple à son maître. On utilise *followerAccessible* pour le dire.
- **créateur** : Avec celle-ci on peut lier un créateur à son œuvre. Nous la nommons *creatorAccessible*.
- **dirigeant non politique** : En utilisant *managerAccessible* on peut lier un chef à l'entité qu'il dirige.
- **locataire** : Comme dans l'exemple "`Barack Obama` est le *locataire* de la `Maison Blanche`" on utilise la relation *tenantAccessible* pour lier une personne à sa résidence. Ce qui donne pour notre exemple :
  Barack Obama $\xrightarrow{tenantAccessible}$ *White House*.
- **héritier** : Avec la relation *heirAccessible* on lie un héritier à son testateur.
- **siège** : Nous définisson la relation *headquatersAccessible* pour lier une première entité à une deuxième pour laquelle elle est son siège.
- **rival** : Avec *rivalAccessible* on pourra lier deux rivaux.
- **companion** : Elle permet de dire que'un tel est companion d'un tel autre. On utilise la relation *companionAccessible*.

Remarques : Il faut d'abord préciser que ces relations sont ni symétriques ni transitives. Nous avons aussi défini une relation inverse pour chacune : *relative*, *capital*, *leader*, *founder*, *follower*, *creator*, *manager*, *tenant*, *heir* , *headquater*, *rival* et *companion*.

2.2 Exemple

Pour illuster les relations entre pivots nous utilisons la *figure 9*. Dans cette figure nous avons quatre pivots :

— La **France** ;
— **Paris** qui est accessible à partir de la **France** en tant que sa capitale ;
— **Île-de-France** qui contient **Paris** ;
— **Cité de Lumière** qui est un synonyme diaphasique de **Paris** ;

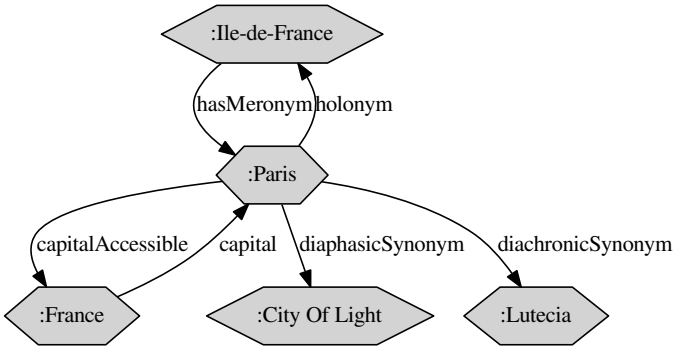

FIGURE 9 – Un exemple de relations de la couche conceptuelle

Voici un autre exemple qui met en jeux les groupes de synonymes. Ici nous avons trois pivots : la **mer rouge**, le **Golfe arabo-persique** et La **mer Érythrée** qui forment un même groupe désigné par **SynSet-40726**. En même temps on précise que c'est la **mer rouge** que l'on utilise le plus souvent ; donc il est canonique pour ce SynSet.

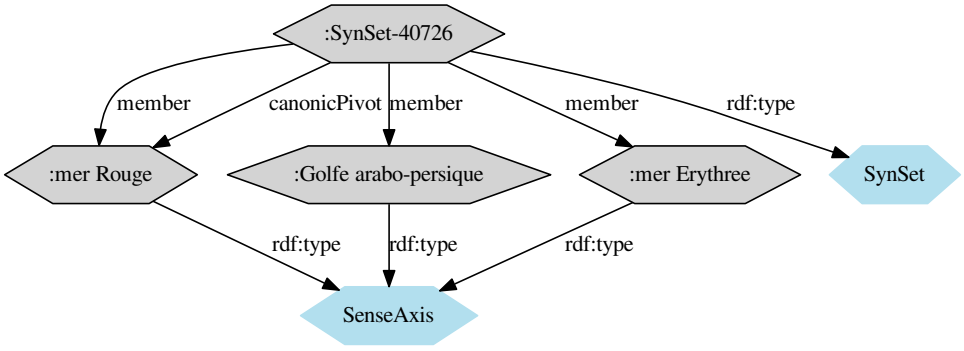

FIGURE 10 – Un exemple de SynSet

3 La partie linguistique

Dans cette partie on retrouve la réalisation des noms propres dans une langue donnée. La projection d'u pivot (niveau conceptuel) donne le (ou les) nom(s)

propre(s) . Regardons les concepts.

3.1 Les concepts

1. **LexicalEntry** : C'est elle qui contient en fait les noms propres et leurs dérivés. Ainsi la projection du nom propre dans la langue choisi donne lieu à une ou plusieurs entrées lexicales. On distingue alors trois catégories d'entrées lexicales :

   (a) **Canonical** : C'est l'élément canonique pour désigner le nom propre dans le cas où ce dernier a plusieurs appelations. Par exemple on considérera *Shanghaï* comme élément canonique pour désigner la ville, alors que chacun des suivants *Changhaï*, *Shanghai* ou *Chang-hai* pouvait faire l'affaire.

   (b) **Alias** : Cet entité sert à représenter toutes les autres entités du nom propre en dehors de celle désignée comme canonique. Dans l'exemple précédent chacun de *Changhaï*, *Shanghai* et *Chang-hai* est un alias pour *Shanghaï*.

   (c) **Derivative** : On le défini récursivement comme tout autre mot que l'on peut dériver d'un nom propre ou d'un dérivé. Il faut comprendre par là que l'on prend aussi en compte les dérivés de dérivé.

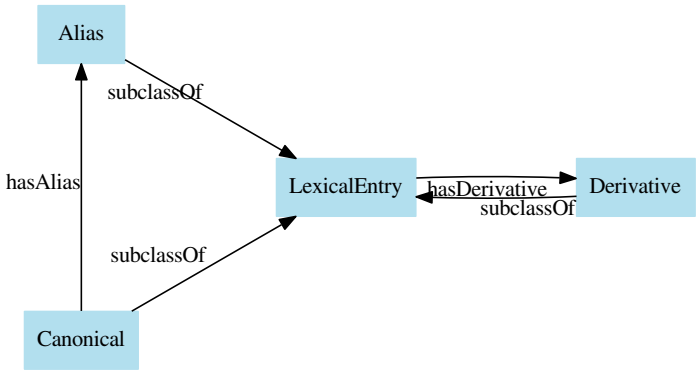

FIGURE 11 – L'entrée lexicale

Comme on le voit dans la *figure* 11 on défini la relaion *hasAlias* entre un `Canonical` et un `Alias`. De la même manière on a la relation *hasDerivative* entre n'importe quelle entrée lexicale et un `Derivative`.
Pour le reste des concepts, on va les présenter par groupe suivant leur association pour déterminer une certaine notion.

3.1.1 La collocation

La collocation consiste à associer des mots à un nom propre pour qu'ils forment un bloc. La *figure* 12 montre le mécanisme que nous avons défini pour la représenter.

FIGURE 12 – La collocation

2. **FunctionalWord** : Les mots fonctionnels sont des éléments qui accompagnent les noms propres. Ils sont utilisés dans la collocation pour permettre par exemple en Français de lier le `FunctionalWord` *en* avec le nom propre *France* pour pouvoir dire "en France".
3. **CollocationCategory** : Cette information reliée à un `FunctionalWord` permet de préciser quel genre de collocation on obtient (déterminant, préposition locative).

Exemple

essayons de voir tout cela à l'aide d'un exemple comme dans la *figure* 13. Dans cet exemple on illustre le fait que le (pays) **Sénégal** a une collocation de catégorie "**préposition locative**" (locative preposition) avec le mot "**au**". Ce qui permet de dire "au Sénégal".

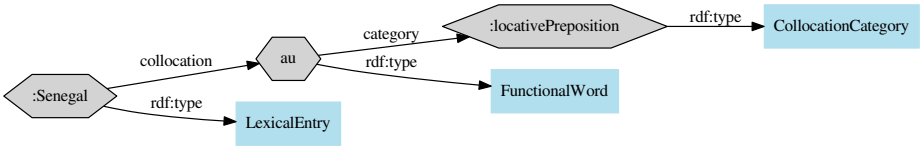

FIGURE 13 – Exemple de collocation

3.1.2 Le contexte

On parle de context quand on essaye d'expliquer le sens d'un nom propre avec d'autres mots du language et/ou en utilisant d'autres noms propres. La *figure* 14 nous fait l'économie des différents concepts et relations utilisés à cet égard :

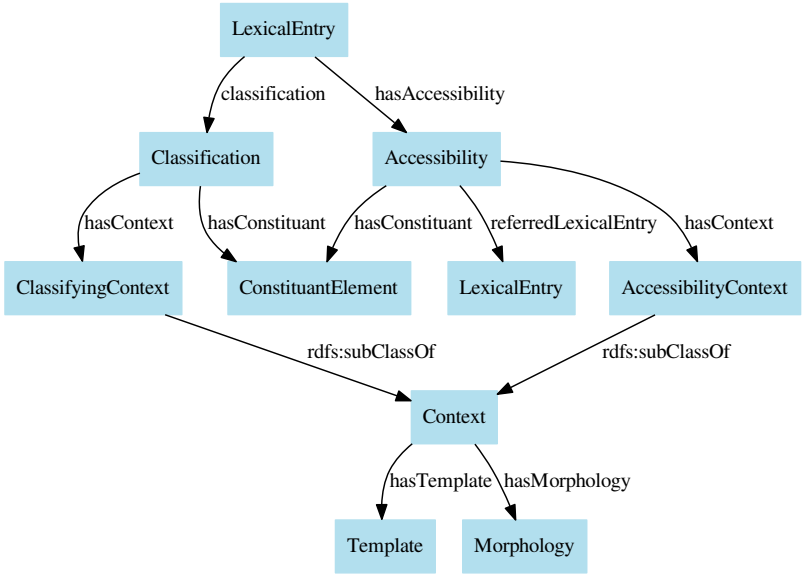

FIGURE 14 – Le contexte

4. **Context** : Le contexte définit une structure externe qui peut venir s'apparenter à une forme du nom propre. Il se définit par deux élément : le patron de production (`Template`) et la morphologie (`Morphology`).
5. **Template** : Cet élément sert à gloser un nom propre. Il se définit donc comme un groupe de mot accompagné d'un paramètre pour le compléter. Par exemple le pattern "La ville de $" où le signe $ sera remplacé par le nom d'une ville (par exemple *Paris* → *La ville de Paris*).
6. **Morphology** : Cet élément vient renseigner sur la partie morphologique et précise le genre et le nombre. On peut l'assoicer à un nom propre ou bien un groupe de mot. On peut par exemple dire que le groupe de mot "*la région de Paris*" est sigulier féminin.
7. **ClassifyingContext** : Cette entité est une spécialisation de `Context`. Elle a donc les mêmes caractéristiques. Seulement elle est exclusivement réservée à une utilisation dans la classification.
8. **AccessibilityContext** : Cette entité est une spécialisation de `Context`. Elle a donc les mêmes caractéristiques. Seulement elle est exclusivement réservée à une utilisation dans l'accessibilité.
9. **Classification** : Cet élement représente en elle la classification c'est à dire le fait de pouvoir faire une glose sur un nom propre. Il est lié à son contexte (`ClassifyingContext`) et à la partie qui participe (`ConstituantElement`) à cette glose pour le cas des noms formés de plusieurs mots. Pour faire simple, on donne seulement le début de la partie à utiliser (premier constituant, deuxième constituant, ...).

10. **Accessibility** : Comme la classification, il sert à paraphraser un nom propre en utilisant un autre nom propre (désigné par *referredLexicalEntry*). Comme la classification, il a son contexte (`AccessibilityContext`), son constituant (`ConstituantElement`) mais aussi le nom propre à utiliser pour la paraphrase.
11. **ConstituantElement** : Dans le cas des noms propres composés de plusieurs mots, cet élément permet d'identifier chacun d'entre ces mots. Nous avons pris la convention de représenter cet élément par son numéro dans l'ordre de succession des mots.

Nous pouvons voir tous ces concepts résumés dans la *figure* 14 ainsi que leurs relations. Il faut aussi préciser que l'alias ou le cnonical peuvent jouer ici le rôle de LexicalEntry et se lier à chacun de ces éléments. On peut voir tout cela à l'aide d'exemples.

Exemples

Dans le premier exemple (*figure* 15) on représente la classification du (fleuve) Sénégal. Cette classification a lieu dans un contexte où on cherche à produire quelque chose du genre "le fleuve..." et qui sera masculin singulier. Et prenant tout le nom propre (commencer par le constituant 1) on aura le résultat "*le fleuve Sénégal*".

Dans le second exemple (*figure* 16) c'est l'accessibilité qui est montrée. Cette dernière aura pour résulat la glose suivante : "*Marie Curie, l'épouse de Pierre Curie*". Pour ce faire on commence par dire qu'elle utilise un patron de conception du genre "l'épouse de..." et que le résulat escompté sera un féminin singulier. Ensuite on dit que c'est tout le nom qui sera utilisé (prendre à partir du constituant 1). Puis Il reste seulement à dire que l'espression sera complétée par son mari, **Pierre Curie**.

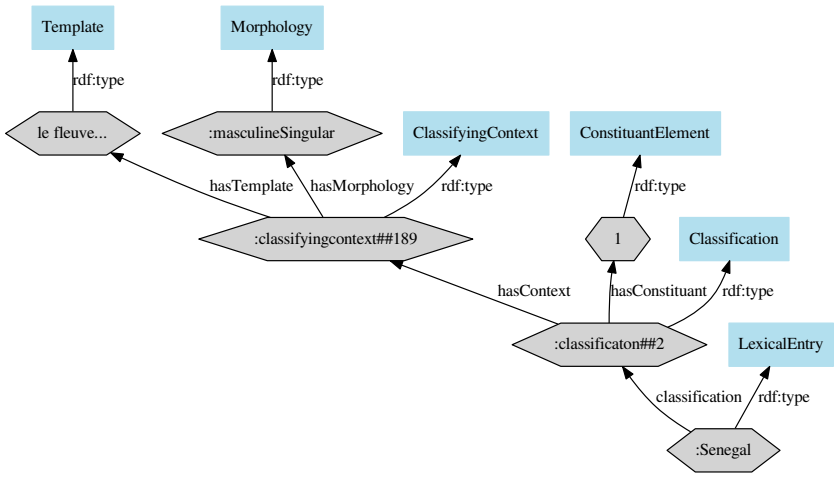

FIGURE 15 – exemple de classification

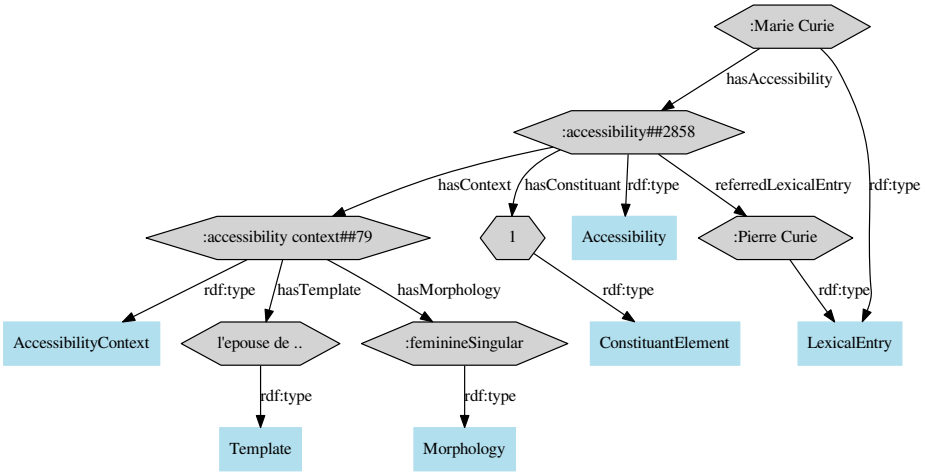

FIGURE 16 – exemple d'accessibilité (linguistique)

3.1.3 L'alias

On parle d'alias quand on peut désigner le même référant par une autre appelation. Tel que montré dans la *figure* 17, l'alias réfère à son Canonical (pour qui il fournit une autre appelation) et précise sa catégorie.

FIGURE 17 – L'alias

12. **Alias** : C'est cet entité qui contient l'alias en question.
13. **AliasCategory** : Il précise la catégorie d'alias que nous avons. Dans la définition de **Prolexbase** les individus qui représentent les catégories sont déja connus.

Pour `Canonical` et `LexicalEntry` se référer aux explications de la *figure* 11.

Exemple

Dans l'exemple suivant (*figure* 18) on a l'information selon laquelle "Marie Curie" peut aussi être appelée "Maria Sklodowska" qui n'est rien d'autre qu'une variante de son nom.

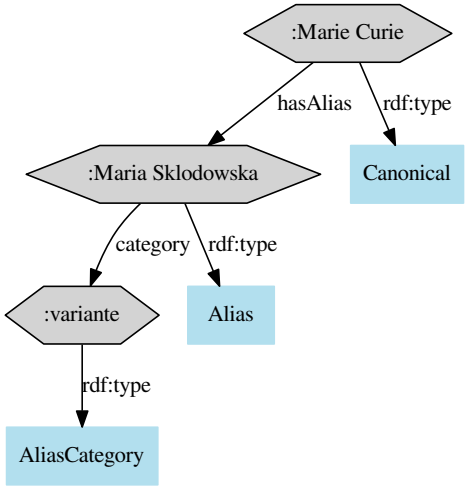

FIGURE 18 – exemple d'alias

3.1.4 La dérivation

On peut dériver d'autres mots à partir des noms propres ou même à partir de leurs dérivés. Là aussi on catégorise les dérivations. La *figure* 19 nous montre comment on la structure.

FIGURE 19 – La dérivation

14. **Derivative** : C'est lui qui contient l'espression du mot dérivé.
15. **DerivativeCategory** : Il sert à catégoiser les dérivés. Toutes les valeurs possibles sont déja connues et chacune d'entre elle sera déclarée comme individu.

    Pour `LexicalEntry` se référer aux explications de la *figure* 11.

    Exemple

    Dans l'exemple illustré dans la *figure* 20 on nous informe que "*jésuite*" est un nom relationnel dérivé du nom propre "*Compagnie de Jésus*".

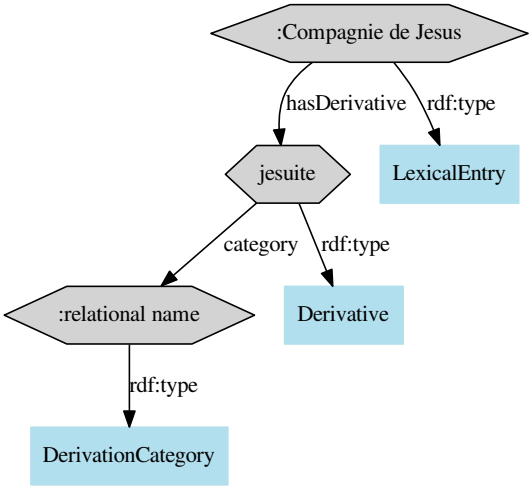

FIGURE 20 – Exemple de dérivation

3.1.5 L'éponymie

On parle d'éponymie quand on rencontre un nom propre dans une autre expression où il est utilisé dans un contexte différent. Et dans ce cas sa traduction ne fait pas forcément référence à lui. Et nous connaissons trois types d'éponymie diffrentes que nous allons présenter.

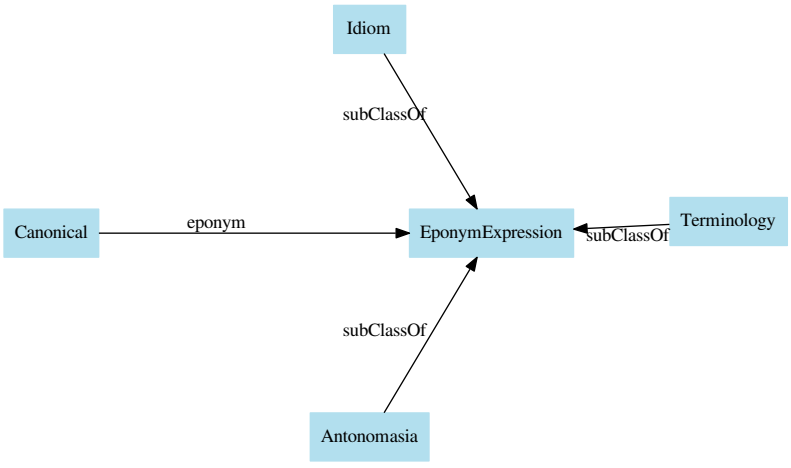

FIGURE 21 – L'éponymie

16. **EponymExpression** : C'est une entité globale et générale qui contient

tous les types d'éponymie. chacun de ces types est donc une spécialisation de ce concept.

17. **Antonomasia** : Dérivé de `EponymExpression`, il est utilisé lorsqu'on désigne un nom commun par un nom propre.
18. **Terminology** : Pareil que le précédent sauf qu'il sert lorsqu'on utilise un nom propre dans la terminologie d'un domaine (e.g *théorème de Pythagore* ou encore *maladie d'Alzheimer*).
19. **Idiom** : Celui-ci est utilisé quand un nom propre participe dans une expression idiomatique comme dans "*Tous les chemins mènent à Rome*".

Exemples

Dans la *figure* 22 on montre trois exemples d'éponymie de formes différentes :
— Le nom de **James Parkinson** utilisé dans la terminologie médicale *maladie de Parkinson* (*Parkinson's* chez les Anglais) ;
— Le nom de la ville de **Rome** qui revient dans l'expression idiomatique *Tous les chemins mènent à Rome* ;
— tartuffe qui s'obtient par l'antonomase du nom propre **Tartuffe**.

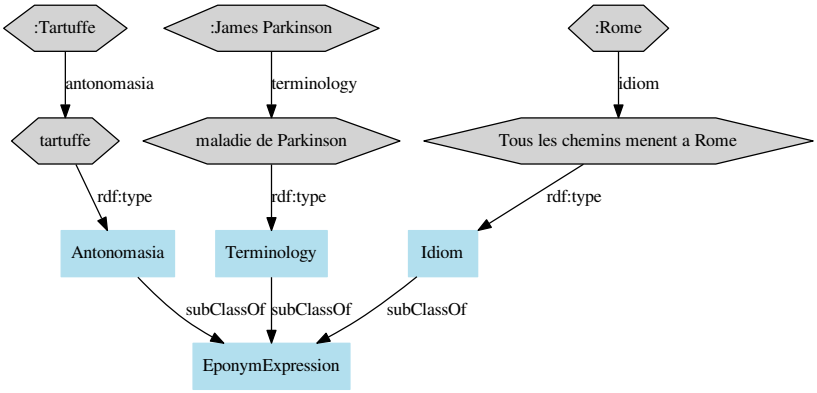

FIGURE 22 – Exemples d'éponymie

3.1.6 Autres informations

Au delà de ces concepts, viennent d'autres qui nous renseignent de plus sur les noms propres. On peut ainsi préciser la langue, ou encore la fréquence d'utilisation dans la langue.

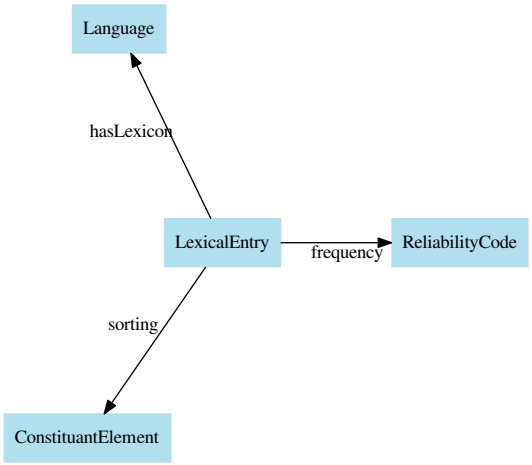

FIGURE 23 – Autres

20. **ReliabilityCode** : Cet entité contient la fréquence d'utilisation d'un nom propre. Pour représenter les différents niveau d'utilisation on réutilise les codes ISO (frequently, rarely and commonly).
21. **Language** : Il représente une langue. Pour cela on utilise le code ISO associé à chaque langue.
22. **ConstituantElement** : Ici il est utilisé pour désigner le constituant qui entre en jeu dans un tri lexicographique. Par exemple pour chercher le nom propre `Georges Washington` dans le dictionnaire, c'est le deuxième constituant `Washington` qu'il faut considérer.

Exemples

Dans la *figure* 24 nous donne des informations qui viennent compléter le niveau linguistique. On peut connaitre ainsi la langue d'un prolexème, sa fréquence d'utilisation et le mot significatif dans le tri d'un dictionnaire. Ansi dans le nom "Etat de Palestine" on regardera avec son troisième constituant qui est "Palestine".

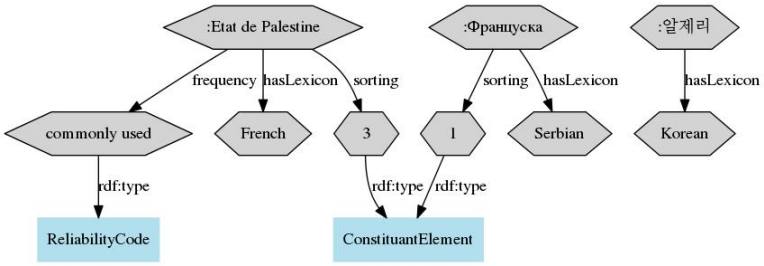

FIGURE 24 – Exemples d'autres informations linguistiques

4 La partie instances

Dans cette dernière couche nous verrons la réalisation de éléments du nom propre dans diffrentes formes. C'est ces instances qui représentent les flexions qui peuvent être en genre ou en nombre. La *figure* 25 montre les différents concepts mis en jeu ainsi que leurs relations.

- **Form** : C'est dans cet élément là que l'on relève l'instence basée sur une entrée lexicale. Il peut représenter alors une forme fléchie de son antécédant. Il y a deux classes qui dérivent de celle-ci :
  - **Lemma** : C'est la forme lemmatique de l'entrée lexicale c'est-à-dire celle utilisée par défaut ;
  - **WordForm** : On y trouve les autres instances de l'entrée lexicale et se voit pour la plus part du temps attacher des informations sur sa morphologie et la partie du discours.
- **PartOfSpeech** : Il représente la partie du discours ou catégorie grammaticale de la forme fléchie. Pour chaque catgorie il y'aura un individu pour le représenter.
- **Morphology** : Nous l'avions présentée dans la partie dédiée aux éléments linguistiques (voir page 13).

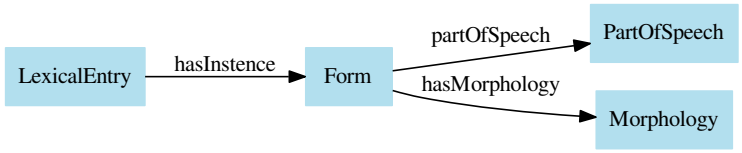

FIGURE 25 – Exemples d'éponymie

Exemples

Dans la *figure* 26 nous avons un exemple d'instance. Il s'agit d'une flexion en féminin singulier du mot '*francilien* qui est dérivé du nom propre "Île de France".

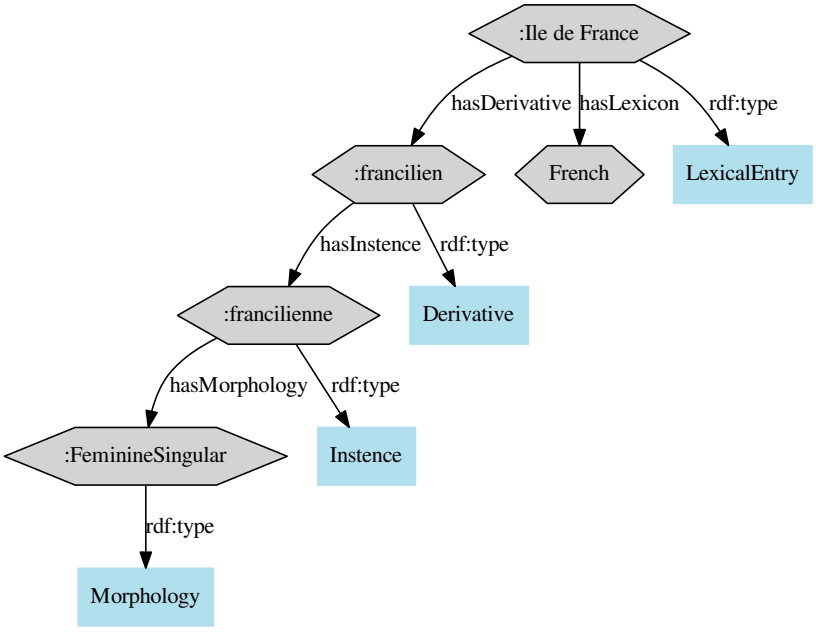

FIGURE 26 – Exemples d'éponymie

Références

[1] Denis MAUREL, Malgorzata SPEDZIA, and Agata SAVARY. Multilingual relational database of proper names : Prolexbase documentation. 2011.