Rapport: Ontologie du Domaine du Financement Public de la Recherche présenté au Professeur Petko Valtchev dans le cadre du cours DIC-9335

Dominique S. Loyer
Université du Québec à Montréal (UQAM)
Département d'informatique
Montréal, QC, Canada
loyer.dominique@courrier.uqam.ca

Abstract

Ce mémoire accompagne l'ontologie OWL 2 DL élaborée afin de modéliser le domaine de la gestion des concours de subventions à la recherche, conformément aux directives fournies pour le cours DIC9335. L'ontologie, conçue en syntaxe Turtle et chargée via Protégé, se propose de capturer les concepts fondamentaux, relations et règles de classification du domaine. Ce document offre une description approfondie du domaine, l'inventaire des classes et propriétés modélisées, ainsi qu'une analyse exhaustive des choix de modélisation relatifs aux types de propriétés, restrictions et axiomes OWL adoptés, incluant la mise en œuvre et la discussion de l'Étape 2 concernant la classification par inférence. L'objectif est de consigner la structure et la logique de l'ontologie développée.

1 Introduction

Ce document décrit la conception et la modélisation d'une ontologie OWL 2 (W3C OWL Working Group, 2012) pour le domaine de la gestion des concours de subventions à la recherche. Ce projet a été réalisé dans le cadre du cours DIC9335 (présenté le 12 avril 2025), en se basant sur l'énoncé fourni ('TravailOnto-e1-SubvRech.pdf'). Le but principal était de développer un modèle formel représentant les entités principales (chercheurs, propositions, comités, évaluations, etc.), leurs relations, ainsi que les règles métier spécifiques, en particulier celles concernant la classification des propositions en catégories (Étape 2 de l'énoncé).

L'ontologie a été créée en utilisant le langage OWL 2 DL, norme du W3C pour la représentation des connaissances sur le Web Sémantique, et éditée en syntaxe Turtle (Beckett et al., 2014). L'outil Protégé (The Protégé Project, 2025) a été utilisé pour le chargement, la visualisation et éventuellement le raffinement du modèle. La méthode générale s'inspire des principes décrits par Noy and McGuinness (2001).

Ce rapport est structuré comme suit : la Section 2 présente une description détaillée du domaine modélisé, en incluant les classes (voir figure 4 en annexe) et propriétés ajoutées. La Section 3 explique en détail les choix de modélisation spécifiques en insistant sur les types et axiomes de propriétés et de classes, ainsi que sur les restrictions utilisées. La Section 4 traite plus particulièrement de l'analyse et de l'implémentation de l'Étape 2 concernant la classification par inférence. La Section 5 conclut le rapport. Les références bibliographiques sont fournies à la fin.

2 Description Enrichie du Domaine

Le champ modélisé concerne le processus de gestion des appels à propositions pour le financement public de la recherche scientifique. Le texte initial décrit un processus typique : une annonce du concours via un appel à propositions, la soumission des propositions par les chercheurs, l'évaluation par des comités spécialisés selon les disciplines, la prise de décision et la communication. Il détaille la structure des comités (président, membres, nombre), les catégories de chercheurs (junior, établi, senior), les critères d'évaluation courants (excellence, potentiel, formation, impacts) et le système de notation (A/Excellent, B/Très Bon, C/Bon). Il spécifie également un processus de recommandation (financement, conditionnel, rejet) basé sur une évaluation sommaire, aboutissant à une classification des propositions en Catégories I, II, et III selon des règles précises basées sur le nombre de notes obtenues.

Lors de la modélisation, les concepts suivants ont été définis, incluant des ajouts ou des précisions par rapport au texte source pour garantir la cohérence et la fonctionnalité du modèle :

2.1 Classes Modélisées

Classes Principales (Explicites ou Fortement Impliquées) :

- `:Concours` [1] : Un concours de subvention spécifique.
- `:AppelAPropositions` : Le document annonçant le concours.
- `:PropositionDeProjet` : Une proposition de recherche soumise.
- `:Chercheur` : Un acteur du processus.
- `:ComiteEvaluation` : Le comité chargé de l'évaluation.
- `:DisciplineDeRecherche` : Le domaine scientifique.
- `:CritereEvaluation` : Les critères utilisés pour l'évaluation.
- `:EvaluationSommaire` : La synthèse des évaluations pour une proposition.
- `:RecommandationComite` : La recommandation issue de l'évaluation sommaire (Financement, Conditionnel, Rejet). Modélisée comme classe contenant des individus pour les types de recommandation.
- `:NoteEvaluation` : Les notes possibles (A, B, C). Modélisée comme classe contenant des individus pour chaque note.

Classes Ajoutées/Précisées pour la Modélisation :

- `:Evaluation` : Classe mère pour regrouper les types d'évaluations.
- `:EvaluationIndividuelle` : Pour représenter l'évaluation par un seul membre.
- `:DecisionFinancement` : Pour représenter la décision finale (distincte de la recommandation).
- `:ChercheurJunior`, `:ChercheurEtabli`, `:ChercheurSenior` : Sous-classes disjointes de `:Chercheur`.
- `:PorteurDeProjet` : Sous-classe de `:Chercheur`.
- `:MembreComite` : Sous-classe de `:Chercheur`.
- `:PresidentComite` : Sous-classe de `:MembreComite` et `:ChercheurSenior`.
- `:StatutProposition` : Classe mère pour les catégories finales.
- `:Categorie_I`, `:Categorie_II`, `:Categorie_III` : Sous-classes disjointes de `:StatutProposition`.
- `:NoteAttribuee` : Classe auxiliaire clé pour lier une note à un critère dans une évaluation. Rationale : Permet de compter les notes par type via des restrictions sur les propriétés associées.
- `:PropositionDeCategorieI`, `:PropositionDeCategorieII`, `:PropositionDeCategorieIII` : Classes définitionnelles (utilisant `owl:equivalentClass`) Les règles de classification de l'Étape 2 visent à permettre au raisonneur de déduire l'inclusion au sein de ces catégories. (et par conséquent au statut final via `:aPourStatut`) basé sur les notes de l'évaluation sommaire.

2.2 Propriétés Modélisées

Propriétés d'Objet : (voir figure 3 en annexe)

- `:annonceConcours` / `:estAnnoncePar` (inverse)
- `:concerneDiscipline`
- `:soumetProposition` / `:estSoumisePar` (inverse, fonctionnelle/inverse fonctionnelle)
- `:aPourAuteur`
- `:estEvalueeParComite` / `:evalueProposition` (inverse)
- `:aPourMembre` / `:estMembreDe` (inverse)
- `:aPourPresident` / `:presideComite` (inverse, fonctionnelle)
- `:assigneEvaluationA`
- `:realiseEvaluationIndividuelle`
- `:concerneProposition` (fonctionnelle)
- `:genereEvaluationSommaire`
- `:estBaseSur`
- `:emetRecommandation`
- `:aPourRecommandation` (fonctionnelle)
- `:aPourNoteCritere` : Lie `:Evaluation` à `:NoteAttribuee`.
- `:concerneCritere` : Lie `:NoteAttribuee` à `:CritereEvaluation` (fonctionnelle).
- `:aValeurNote` : Lie `:NoteAttribuee` à

1. J'ai indiqué la modélisation en format de style Type Writer pour faciliter la lecture

:NoteEvaluation (fonctionnelle).
- :determineStatut : Lie :RecommandationComite à :StatutProposition (fonctionnelle).
- :aPourStatut : Lie :PropositionDeProjet à :StatutProposition (fonctionnelle).

Propriétés de Données : (voir figure 2 en annexe)
- :dateLimiteSoumission (fonctionnelle, range xsd:date)
- :dateReponseAttendue (fonctionnelle, range xsd:date)
- :titreProposition (fonctionnelle, range xsd:string)
- :nomChercheur (range xsd:string)

3 Choix de Modélisation

Cette section détaille les choix effectués pour représenter les éléments clés du domaine en OWL 2 (W3C OWL Working Group, 2012), en utilisant les vocabulaires RDF (Cyganiak et al., 2014) et RDFS (Guha and Brickley, 2014), et en mettant l'accent sur les aspects demandés dans l'énoncé du travail.

3.1 Types de Propriétés (Caractéristiques)

Plusieurs propriétés ont été déclarées avec des caractéristiques spécifiques (*property characteristics*) pour enrichir la sémantique et permettre des inférences ou des contrôles de cohérence : (Afin de rendre les classes définitionnelles disjointes, j'ai explicitement déclaré que :PropositionDeCategorieI, :PropositionDeCategorieII, et :PropositionDeCategorieIII sont disjointes entre elles (elles étaient déjà implicitement des sous-classes de :PropositionDeProjet via leurs définitions owl :equivalentClass). Les caractéristiques des proprités :

Fonctionnelles (owl:FunctionalProperty) : Garantit qu'un individu a au plus une valeur pour la propriété. Appliqué à : :soumetProposition, :aPourPresident, :concerneProposition, :aPourRecommandation, :concerneCritere, :aValeurNote, :determineStatut, :aPourStatut, :dateLimiteSoumission, :dateReponseAttendue, :titreProposition. Le choix de rendre :aPourNoteCritere non fonctionnelle permet à une évaluation d'avoir plusieurs notes (via plusieurs instances de :NoteAttribuee), tandis que :concerneCritere et :aValeurNote sont fonctionnelles car une :NoteAttribuee lie un seul critère à une seule valeur.

Inverse Fonctionnelles (owl:InverseFunctionalProperty) : Garantit qu'une valeur donnée n'est associée qu'à un seul sujet. Appliqué à : :estSoumisePar (une proposition ne peut être soumise que par une seule personne).

Inverses (owl:inverseOf) : Définies pour les paires : :annonceConcours / :estAnnoncePar, :soumetProposition / :estSoumisePar, :estEvalueeParComite / :evalueProposition, :aPourMembre / :estMembreDe, :aPourPresident / :presideComite.

Autres Caractéristiques : Aucune propriété n'a été déclarée transitive, symétrique, etc., car cela ne semblait pas pertinent pour le domaine décrit.

3.2 Restrictions de Propriétés

Les restrictions (owl:Restriction) sont utilisées pour définir des classes de manière plus précise, souvent en contraignant les valeurs ou le nombre de valeurs pour une propriété donnée :

Restrictions de Cardinalité : (voir figure 5 en annexe) Utilisées sur la classe :ComiteEvaluation pour spécifier le nombre exact de présidents (owl:cardinality "1"^^xsd:nonNegativeInteger sur :aPourPresident) et la fourchette du nombre de membres (owl:minCardinality "4"^^xsd:nonNegativeInteger et owl:maxCardinality "8"^^xsd:nonNegativeInteger sur :aPourMembre, en incluant le président dans le compte total pour simplifier). Également utilisées intensivement dans les axiomes owl:equivalentClass pour les catégories de propositions afin de compter le nombre de notes spécifiques requises (ex : owl:minCardinality "3"^^xsd:nonNegativeInteger sur une restriction filtrant les :Note_A pour :PropositionDeCategorieI).

Restrictions de Valeur : `owl:allValuesFrom` est utilisé sur `:ComiteEvaluation` pour restreindre le type des membres (`:aPourMembre`) à l'union (`owl:unionOf`) de `:ChercheurSenior` et `:ChercheurEtabli`. Également utilisé dans les axiomes finaux liant `:PropositionDeCategorieX` à `:Categorie_X` via `:aPourStatut`. `owl:someValuesFrom` est utilisé de manière centrale dans les axiomes `owl:equivalentClass` pour exprimer des conditions existentielles. Par exemple, une proposition est dans une catégorie si elle a *au moins une* (`owl:someValuesFrom`) évaluation sommaire qui, elle-même, a *au moins une* (`owl:someValuesFrom`) note attribuée dont la valeur (`:aValeurNote`) est une note spécifique (`:Note_A`). `owl:hasValue` est utilisé dans les restrictions les plus internes des axiomes de catégorie pour spécifier que la valeur de la propriété `:aValeurNote` doit être un individu spécifiques (ex : `:Note_A`, `:Note_B`, ou `:Note_C`).

3.3 Axiomes de Classes

Hiérarchie (`rdfs:subClassOf`) : Définit la structure taxonomique (voir Figure I en annexe) de base de l'ontologie (ex : `:ChercheurJunior` est un `:Chercheur`).

Disjonction (`owl:AllDisjointClasses`) : Utilisé pour déclarer que les catégories de chercheurs (`:ChercheurJunior`, `:ChercheurEtabli`, `:ChercheurSenior`) sont mutuellement exclusives, de même que les catégories de propositions (`:Categorie_I`, `:Categorie_II`, `:Categorie_III`). Cela empêche un individu d'appartenir à plusieurs de ces classes simultanément et aide le raisonneur à détecter les incohérences. Ainsi, en lançant le raisonneur Pellet lors du premier test, j'ai constaté que la catégorie II comportait une incohérence (voir Figure 4 en annexe). J'ai pris la décision d'utiliser un autre raisonneur, HermiT 1.4.3.456, très robuste, complet pour OWL 2 DL et préférable pour avec les axiomes complexes, et activement maintenu. C'est un choix judicieux pour une contre-vérification. J'ai également testé le raisonneur FaCT++ 1.6.5, mais il ne supporte pas la balise xsm :date que j'ai utilisée. Quant à ELK 0.6.0 : Il est très rapide, moins expressif, mais optimisé davantage pour le profil OWL 2 EL. Ainsi, il pourrait me donner des résultats incomplets.

Combinaisons Booléennes : `owl:intersectionOf` est essentiel pour les définitions `owl:equivalentClass` des catégories, permettant de combiner plusieurs conditions (ex : être une `:PropositionDeProjet` ET satisfaire une restriction sur l'évaluation ET être le complément d'une autre catégorie). `owl:unionOf` est utilisé dans une restriction sur `:ComiteEvaluation` pour définir les types de membres autorisés. `owl:complementOf` est utilisé dans les définitions des catégories II et III pour exclure explicitement les propositions appartenant déjà à une catégorie supérieure.

Classes Équivalentes (`owl:equivalentClass`) : Au cœur de l'implémentation de l'Étape 2. Les classes `:PropositionDeCategorieI`, `:PropositionDeCategorieII`, et `:PropositionDeCategorieIII` sont définies comme étant logiquement équivalentes à des descriptions de classes complexes basées sur les restrictions de cardinalité (`owl:minCardinality "X"^^xsd:nonNegativeInteger`) et de valeur (`owl:hasValue`) des notes obtenues via l'évaluation sommaire. Cela permet au raisonneur OWL d'inférer automatiquement l'appartenance d'une proposition à l'une de ces classes définitionnelles si elle satisfait les conditions.

3.4 Axiomes de Propriétés

Inversion (`owl:inverseOf`) : Le principal axiome de propriété utilisé, pour lier les paires de propriétés réciproques (ex : `:annonceConcours` / `:estAnnoncePar`). Cela permet au raisonneur d'inférer une relation si son inverse est connue.

Autres : Aucun axiome de disjonction de propriétés (`owl:propertyDisjointWith`) ou de chaîne de propriétés (`owl:propertyChainAxiom`) n'a été défini dans ce modèle.

4 Implémentation et Test de la Classification (Étape 2)

L'Étape 2 de l'énoncé se concentre sur la modélisation et la vérification des règles de classification

des propositions en catégories I, II et III, basées sur les notes de l'évaluation sommaire.

4.1 Analyse de Cohérence des Règles

Les règles de classification fournies sont les suivantes :

— Catégorie I : $\geq$ 3 notes Excellent (A).
— Catégorie II : $\geq$ 2 notes Très Bon (B) ET $\geq$ 1 note Excellent (A).
— Catégorie III : $\geq$ 2 notes Bon (C).

Cohérence : Les règles semblent cohérentes dans le sens où elles ciblent des combinaisons de notes primaires différentes (A pour I, B+A pour II, C pour III). Grâce aux axiomes `owl:complementOf` ajoutés dans les définitions OWL (voir section suivante), une proposition satisfaisant les critères de la Catégorie I ne sera pas classée en Catégorie II ou III, et une proposition de Catégorie II ne sera pas classée en Catégorie III, assurant une classification unique selon la priorité implicite I > II > III.

Cas Contre-intuitifs ou Non Couverts : Le principal point faible de ces règles est qu'elles ne sont **pas exhaustives**. De nombreuses combinaisons de notes ne mènent à aucune classification :

— Moins de 3 A, et pas assez de B ou C (ex : 2 A, 1 B, 1 C).
— Moins de 2 B ou moins de 1 A pour la règle II, et moins de 2 C (ex : 1 A, 1 B, 2 C).
— Moins de 2 C, et pas assez de A ou B pour les autres règles (ex : 1 A, 1 C).

Une proposition avec, par exemple, 2 notes "Excellent" ne serait classée dans aucune catégorie. De même, une proposition avec 1 "Excellent" et 3 "Bon" ne correspond à aucune règle. L'ontologie actuelle reflète ces règles telles quelles ; une proposition ne satisfaisant aucune des conditions `owl:equivalentClass` ne sera simplement pas inférée comme appartenant à `:Categorie_I`, `:Categorie_II`, ou `:Categorie_III`. Il faudrait potentiellement définir une catégorie par défaut (ex : "Non Classé" ou "Autre") ou affiner les règles pour couvrir tous les cas si nécessaire dans un contexte réel.

4.2 Modélisation OWL des Règles

Comme détaillé dans la Section 3.3, les règles de classification ont été implémentées en utilisant des axiomes `owl:equivalentClass` pour les classes définitionnelles `:PropositionDeCategorieI`, `:PropositionDeCategorieII`, et `:PropositionDeCategorieIII`.

Le mécanisme repose sur :

1. La navigation depuis une `:PropositionDeProjet` vers son `:EvaluationSommaire` via la propriété `:aPourEvaluationSommaire` (utilisant `owl:someValuesFrom`).
2. La navigation depuis l'`:EvaluationSommaire` vers les différentes `:NoteAttribuee` via la propriété `:aPourNoteCritere`.
3. Le filtrage et le comptage des `:NoteAttribuee` en fonction de la valeur de leur note (liée via `:aValeurNote`) en utilisant des restrictions imbriquées avec `owl:hasValue` (pointant vers les individus `:Note_A`, `:Note_B`, `:Note_C`) et `owl:minCardinality`.
4. La combinaison de ces conditions et des exclusions (`owl:complementOf`) via `owl:intersectionOf`.
5. La liaison finale entre l'appartenance à une classe définitionnelle (ex : `:PropositionDeCategorieI`) et l'attribution du statut final (ex : `:Categorie_I`) via un axiome `rdfs:subClassOf` utilisant une restriction `owl:allValuesFrom` sur la propriété `:aPourStatut`.

Ce choix de modélisation permet d'encoder directement la logique des règles dans la structure de l'ontologie, rendant la classification possible par un raisonneur OWL standard.

4.3 Méthodologie de Test par Inférence

Pour vérifier que la modélisation des règles fonctionne comme prévu, la méthodologie suivante doit être appliquée dans Protégé :

1. **Création d'Individus Test :**
   — J'ai créé une instance (individu) de `:PropositionDeProjet` pour chaque catégorie cible (I, II, III) et potentiellement pour des cas non couverts. Par exemple : `:propTestCatI`, `:propTestCatII`, `:propTestCatIII`, `:propTestNonClassee`.
   — Pour chaque proposition test, j'ai aussi créé une instance de

:EvaluationSommaire (ex : :evalSommaireTestCatI) et j'ai lié les deux via la propriété :aPourEvaluationSommaire (assertion à ajouter sur l'instance de proposition).

2. **Création des Notes Attribuées :**
   - Pour chaque :EvaluationSommaire test, j'ai créé les instances de :NoteAttribuee nécessaires pour satisfaire (ou non) les conditions d'une catégorie. Qui plus est, pour :evalSommaireTestCatI, j'ai procédé à l'élaboration ici de 3 instances de :NoteAttribuee.
   - et liées chaque :EvaluationSommaire à ses :NoteAttribuee via la propriété :aPourNoteCritere.
   - Pour chaque :NoteAttribuee, il faut définir ses propriétés :
     - :concerneCritere : J'ai lié une instance de :CritereEvaluation (J'ai utitlisé :Critere_ExcellenceDossier). Le critère exact importait peu pour tester la logique de classification basée sur les notes seules, mais il a fallu en lier un.
     - :aValeurNote : Liaison à l'instance de :NoteEvaluation appropriée (:Note_A, :Note_B, ou :Note_C). C'est cette liaison qui est cruciale pour les règles.
   - Exemple pour :evalSommaireTestCatI : créer :noteA1, :noteA2, :noteA3. Pour :noteA1, ajouter les assertions :concerneCritere :Critere_ExcellenceDossier et :aValeurNote :Note_A. Faire de même pour :noteA2 et :noteA3 (en liant potentiellement à d'autres critères pour plus de réalisme, mais en liant toujours à :Note_A). Lier ensuite :evalSommaireTestCatI à ces trois notes via :aPourNoteCritere.
   - Créer des jeux de notes correspondants pour les tests des Catégories II, III et des cas non classés.
3. **Lancement du Raisonneur :**
   - J'ai essayé HermiT et Pellet.
   - (classification, vérification de cohérence).
4. **Vérification des Inférences :**
   - Sélection de chaque individu test de :PropositionDeProjet (ex : :propTestCatI).
   - Examination de la section "Types" (ou "Class Hierarchy (inferred)") dans la description de l'individu.
   - Vérification pour voir si le raisonneur a correctement inféré l'appartenance à la classe de statut attendue (:Categorie_I, :Categorie_II, ou :Categorie_III) en plus de son type déclaré :PropositionDeProjet. Note : Pour :propTestNonClassee, aucune de ces classes ne devrait être inférée.

Cette procédure a permis de valider empiriquement que les axiomes owl:equivalentClass et les restrictions de cardinalité ont fonctionnés comme prévu et que le raisonneur a classé correctement les propositions selon les règles métier encodées dans l'ontologie. Par ailleurs, après le test avec le raisonneur Pellet j'ai obtenu une contradiction dans la classe : Proposition de catégorie II. Ainsi, cette contradiction logique dans les axiomes de l'ontologie telle qu'elle est écrite, qui rend la classe :PropositionDeCategorieII intrinsèquement insatisfaisable, c'est-à-dire qu'aucune instance ne peut logiquement exister selon les règles définies.

5 Conclusion

L'ontologie OWL élaborée fournit une représentation formelle du domaine de la gestion des subventions de recherche, telle que spécifiée dans la description initiale et enrichie au cours de la modélisation. Elle capture les entités principales, leurs relations et attributs, ainsi que les contraintes structurelles (composition des comités, types de chercheurs). L'emploi d'axiomes owl:equivalentClass fondés sur des restrictions de cardinalité et de valeur permet notamment de mettre en œuvre la logique de classification automatique des propositions en catégories I, II ou III, démontrant ainsi la

capacité de l'ontologie à supporter le raisonnement inférentiel requis par la deuxième étape du travail. Les choix de modélisation, tels que l'introduction de la classe `:NoteAttribuee`, visent à garantir la cohérence et l'expressivité logique du modèle OWL. L'analyse des règles de classification a révélé qu'elles ne sont pas exhaustives, observation qui est reflétée dans le modèle actuel mais qui pourrait nécessiter un affinage dans un cadre d'application pratique. Les tests par inférence décrits permettent de valider le comportement du modèle vis-à-vis des règles implémentées.

References

Allemang, D., Hendler, J. A., and Gandon, F. (2020). *Semantic Web for the Working Ontologist : Effective Modeling for Linked Data, RDFS, and OWL*. ACM Press / Morgan & Claypool Publishers, New York, NY, USA / San Rafael, CA, USA, third edition.

Beckett, D., Berners-Lee, T., Prud'hommeaux, E., and Carothers, G. (2014). RDF 1.1 Turtle : Terse RDF Triple Language. W3C Recommendation.

Berners-Lee, T. (2010). Long live the Web. *Scientific American*, 303(6) :80–85.

Berners-Lee, T., Weitzner, D. J., Hall, W., O'Hara, K., Shadbolt, N., and Hendler, J. A. (2006). A framework for web science. *Foundations and Trends® in Web Science*, 1(1) :1–130.

Cyganiak, R., Wood, D., and Lanthaler, M. (2014). RDF 1.1 Concepts and Abstract Syntax. W3C Recommendation.

Guha, R. V. and Brickley, D. (2014). RDF Schema 1.1. W3C Recommendation.

Hall, W. (2011). The ever evolving Web : The power of networks. *International Journal of Communication*, 5 :651–664.

Hendler, J., Shadbolt, N., Hall, W., Berners-Lee, T., and Weitzner, D. (2008). Web science : An interdisciplinary approach to understanding the Web. *Communications of the ACM*, 51(7) :60–69.

Hitzler, P., Krötzsch, M., and Rudolph, S. (2009). *Foundations of Semantic Web Technologies*. Chapman and Hall/CRC, Boca Raton, FL, USA.

Noy, N. F. and McGuinness, D. L. (2001). Ontology development 101 : A guide to creating your first ontology. Stanford Knowledge Systems Laboratory Technical Report KSL-01-05 and Stanford Medical Informatics Technical Report SMI-2001-0880.

The Protégé Project (2025). Protégé Ontology Editor. Website and Software.

W3C OWL Working Group (2012). OWL 2 Web Ontology Language Document Overview (Second Edition). W3C Recommendation.

Autre bibliographie et médiagraphie

Tout le matériel vus dans le cadre du cours sciences du web DIC-9335 donné par le professeur Petko Valtchev.
https://protege.stanford.edu/publications/ontology_development/ontology101-noy-mcguinness.html[consulté le 11-12 avril 2025]

Plan

A Figures Annexes

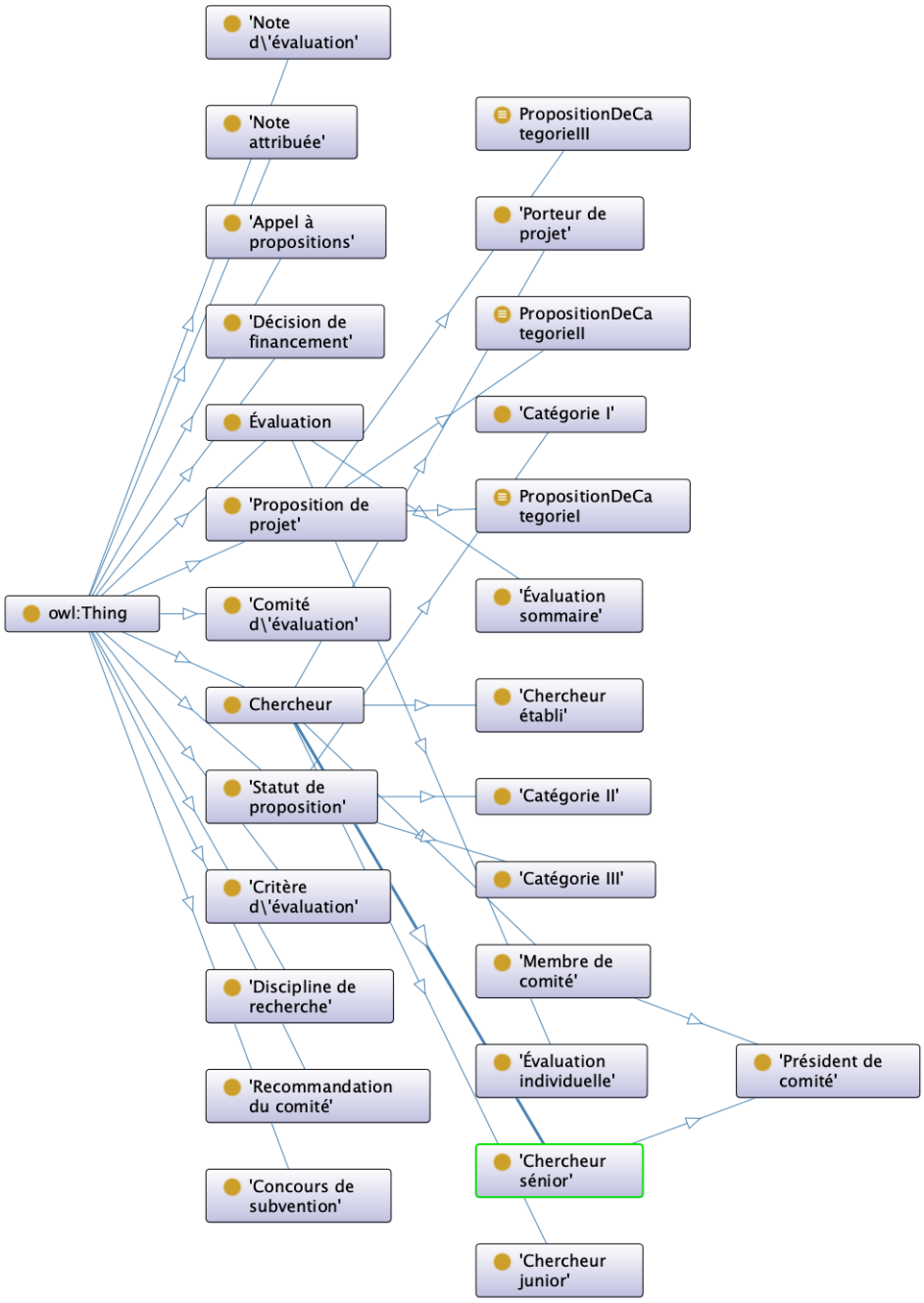

FIGURE 1 – Taxonomie de l'ontologie de recherche.

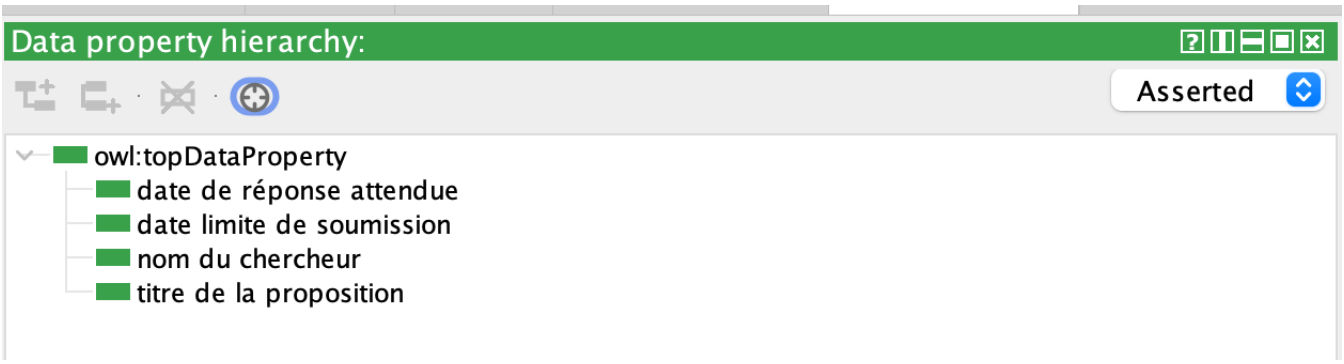

FIGURE 2 – Propriétés des données

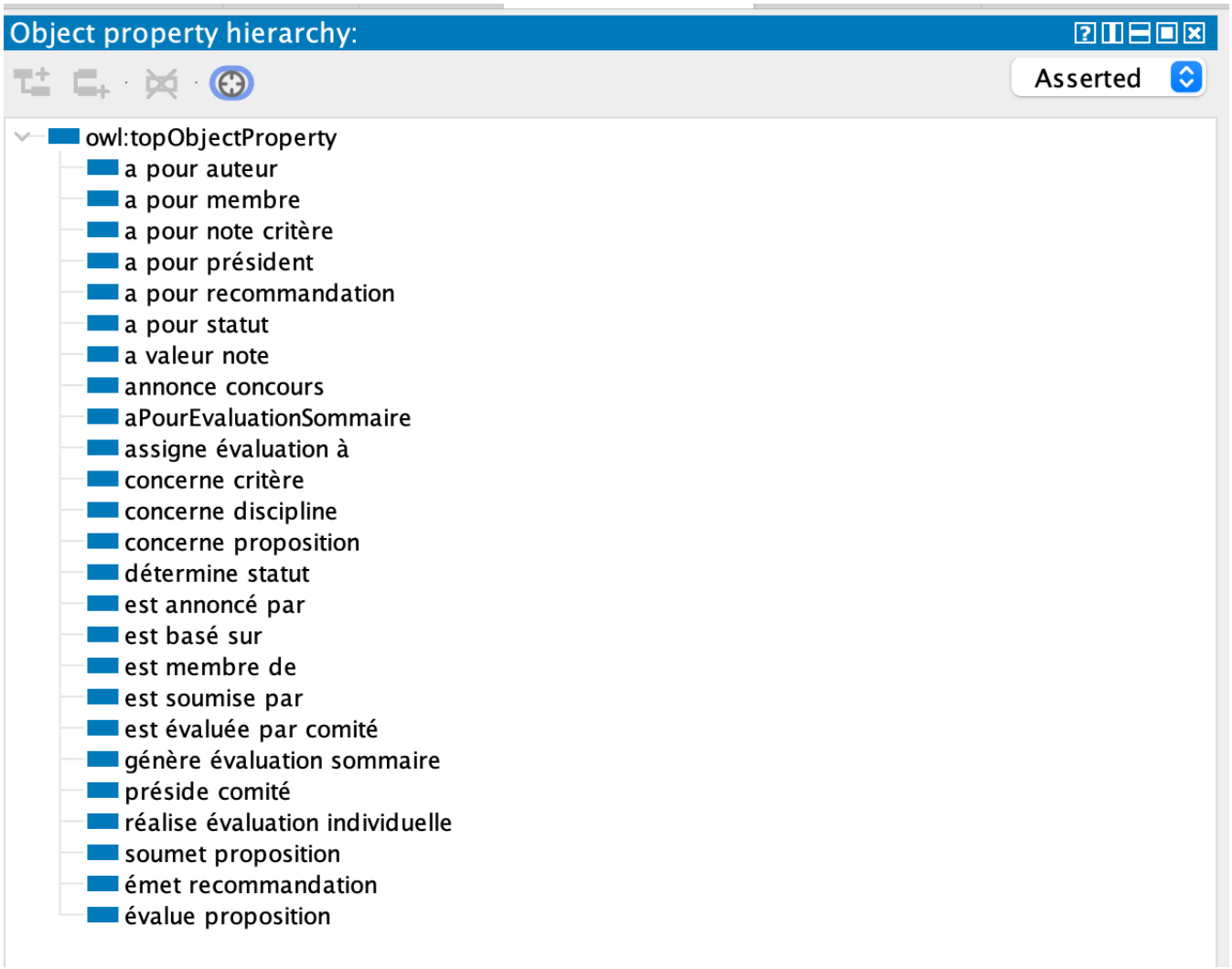

FIGURE 3 – Propriétés des objets

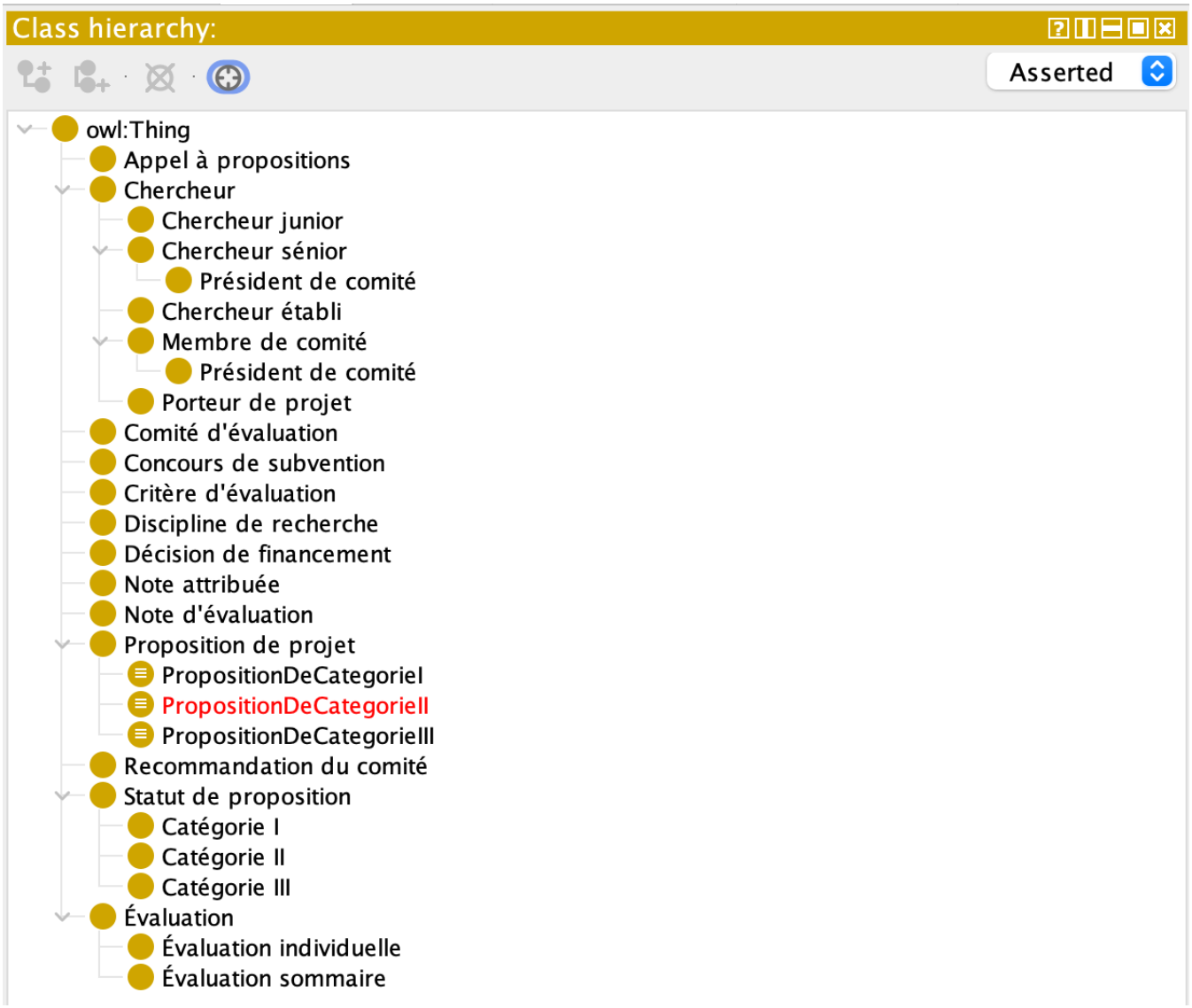

FIGURE 4 – Propriétés des classes

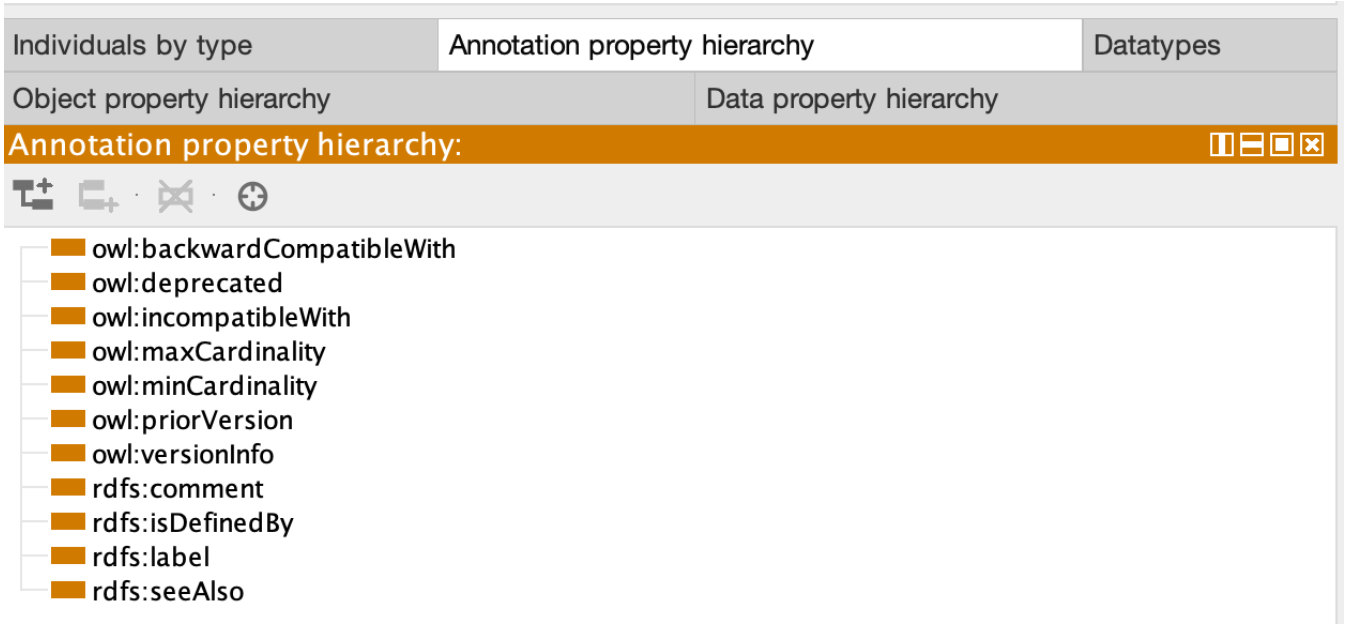

FIGURE 5 – Propriétés des annotations

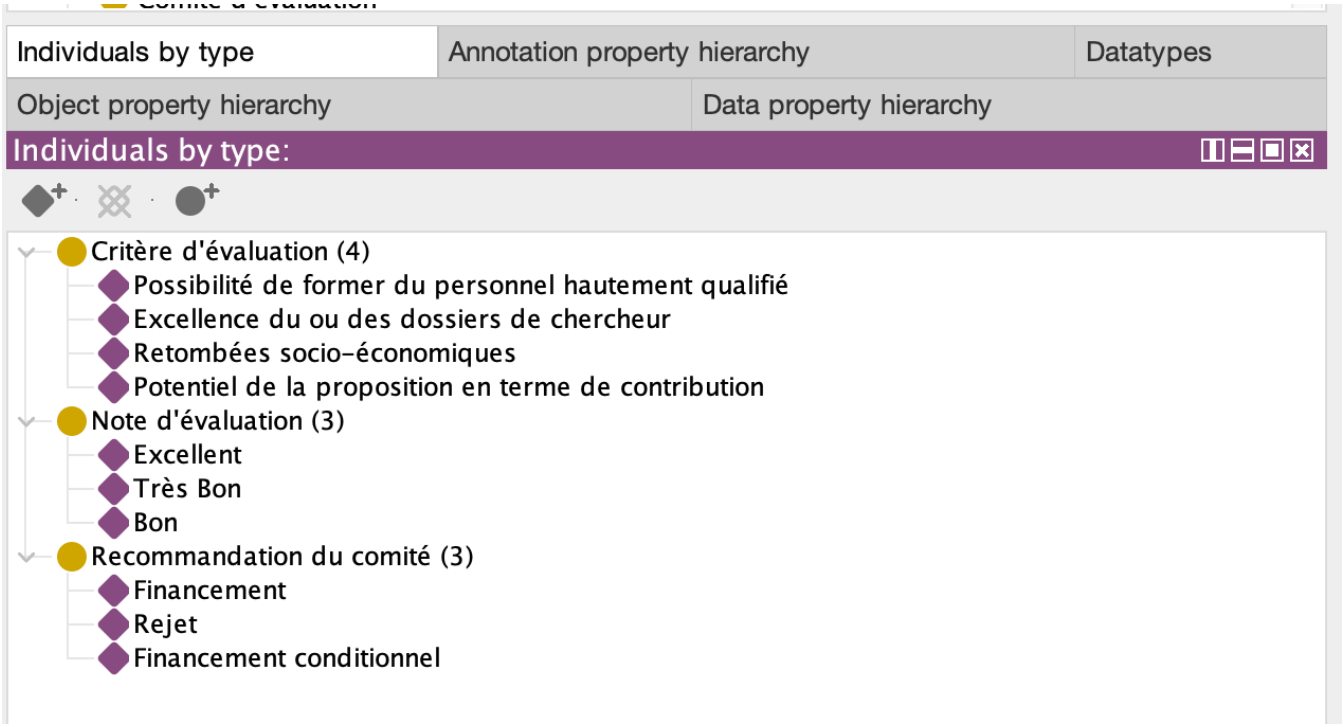

FIGURE 6 – Types d'individus