Lab RDFS entailment

Premierement nous avons essayé de reperer à l’œil ligne par ligne les anomalies qui peuvent être réctifiés par des construct directement dans le fichier. C’est notament comme cela que nous avons su que Sophie n’était pas une ressource qui existe et donc nous l’avons crée. La deuxieme méthode était de lancer des requêtes SPARQL basiques pour identifier les differences entre avec RDFS et sans RDFS. Enfin nous avons véiller à ce que les 11 règles de la recommandation soient utilisées. Pour cela nous avon réécrit en language naturel les 11 régles avant de les implementer ce qui nous a grandement facilité le travail :

pour toute propriété creer subpropriété la meme (régle 6)

pour toute propriété qui a est une sous propriété d’une autre ces deux propriétés relient les même couples sujets objets (régle 7)

pour toute classe, celle-ci est une sous classe de rdfs:Resource(régle 8)

si une ressource prend un type qui lui meme est une sous classe alors lui attribuer la sous classe et la classe (régle 9)

toute classe est sous classe d'elle même (régle 10)

identifier sous classes sur 3 étages et donner subclassof entre 1er et troisieme (régle 11)

Passons maintenant aux construct que nous avons crée spécialement pour les données fournies dans humans.rdf et humans.rdfs. Cela correspond à écrire avec un construct les triplets qui différents entre avec et sans rdfs activé :

*(Si certaines queries ne sont pas devellopées ici c’est parce qu’il n’y a pas de différences)*

Concernant la query 16 : Laura n’est pas de type femme(rdfs activé ou desactivé ne change rien). On rectifie avec la construct 13.

Concerant les queries de 17 à 19 Laura est écrite plusieurs fois dans le fichier avec à chaque fois un des types donc activer rdfs ou pas donne le même résultat.

Cela se remarque bien avec les queries de 21 à 24 avec Gaston qui lui si on lui donne un type autre que Researcher on ne trouve aucun résultat (si rdfs desactivé) à part pour Man. On rectifie avec les construct 14 et 15 (comme régles 11 et 9).

C’est aussi confirmé par les queries de 25 à 28 avec David qui sort pour les types researcher et Person. On rectifie avec les construct de 16 et 17 (comme régles 11 et 9).

Enfin nous avons ajoutés des constructs afin de rajouter des informations au graph

Construct 18 : Toute personne qui a un parent a pour autre parent l’époux du premier.

Si x hasMother ou hasFather y et z hasSpouse x alors x hasFather y.

Construct n19 : Tout ami ou epoux d’un ami est aussi mon ami.

Si x hasfriend y et y hasfriend ou hasSpouse z alors x hasfriend z ( à peu près comme régle 7).

Construct n 20 : Creer type adulte si >18.

Construct n 21 : Pour être un epoux(se) il faut être un ami

*(Disclaimer : ceci sont des exemples et ne réprensentent pas forcément la réalité)*