Sorbonne Université

**Rapport Projet LRC DEMONSTRATEUR POUR LA LOGIQUE ALC EN PROLOG**

Décembre 2023



**Auteurs :**

HADBI Imane

…

# **Introduction**

Ce projet consiste en la conception d'un démonstrateur en Prolog exploitant l'algorithme des tableaux pour la logique ALC. Structuré en trois parties, chacune revêt une importance particulière pour atteindre les objectifs globaux. La première partie se focalise sur la garantie de la validité syntaxique et sémantique des éléments de la logique ALC. La deuxième partie concerne la saisie des propositions à démontrer, tandis que la troisième et dernière partie implique la construction du démonstrateur. À ce stade, l'algorithme des tableaux sera intégré, permettant au système de générer des preuves et de démontrer la validité des énoncés logiques introduits par l'utilisateur.

# **Implémentation de la première partie**

## **Liste des Concepts, Instances et Rôles**

Nous avons développé des prédicats (**listTBox**, **listAboxInst**, **listAboxRole**) pour générer des listes des concepts, instances et rôles de la TBox et de l'ABox respectivement.

## **Vérification Syntaxique et Sémantique**

Pour assurer la validité syntaxique et sémantique, des prédicats tels que **concept/1**, **concept/2**, et **concept/3** ont été définis pour vérifier la cohérence des concepts, instances et rôles. La vérification syntaxique et sémantique est réalisée à l'aide du prédicat **verifier\_concept/3**, qui parcourt les éléments de la TBox et de l'ABox pour s'assurer de leur conformité avec les règles définies. En cas d'incohérence, le système signale les concepts invalides, assurant ainsi une syntaxe et une sémantique correctes.

## **Détection de l'Auto-référence**

Le prédicat **autoref/2** a été élaboré pour traiter divers scénarios d'auto-référence, reconnaissant tant les auto-références directes que les auto-références indirectes. Il identifie l'auto-référence directe lorsqu'un concept apparaît dans sa propre définition (circularité directe) et gère l'auto-référence indirecte en examinant les équivalences entre concepts et en parcourant récursivement les définitions des concepts équivalents pour confirmer l'apparition du même concept. Le prédicat **pas\_autoref/1** vérifie l'absence d'auto-référence dans la liste des couples (C, E) de la T-Box. Il utilise le prédicat autoref/2 pour chaque couple, assurant ainsi l'absence d'auto-référence dans toute la liste. Le prédicat **verifier\_auto\_reference/1** prend en entrée la T-Box et effectue une vérification complète. S'il ne détecte aucune auto-référence, il affiche un message indiquant que la T-Box n'est pas auto-référente. En revanche, s'il identifie une auto-référence, il signale une erreur et provoque un échec.

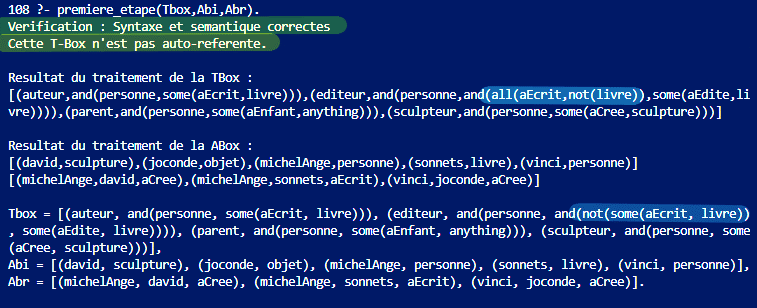
## **Traitement de la T-Box et de l'A-Box**

Le traitement des T-Box et A-Box repose sur le prédicat **remplace/2**, qui réalise un remplacement récursif des identificateurs de concepts complexes. La T-Box est traitée par **traitement\_Tbox/2**, où les expressions conceptuelles équivalentes à des concepts complexes sont remplacées par des identificateurs atomiques et mises sous forme normale négative. De même, **traitement\_Abox/2** opère sur l'A-Box, substituant les identificateurs complexes par leurs définitions de la T-Box simplifiée, suivies d'une mise sous forme normale négative. L'orchestration complète est assurée par **traitement\_Tbox\_Abox/3**, affichant les résultats du traitement.

## **Exécution de la Première Étape**

Le prédicat **premiere\_etape/3** exécute la première étape du projet, englobant toutes les vérifications et traitements nécessaires. Il débute par la validation syntaxique et sémantique des éléments, assurée par le prédicat **verifier\_concept/3**. En suivant, il s'assure de l'intégrité de la T-Box en vérifiant l'absence d'auto-référence à l'aide de **verifier\_auto\_reference/1**. Enfin, le traitement de la T-Box et de l'A-Box est effectué par le prédicat **traitement\_Tbox\_Abox/3**, présentant ainsi des résultats prêts pour la suite du projet.

L'exécution de la première étape du projet a produit les résultats suivants :



On observe que suite à la vérification syntaxique, sémantique et la détection d'auto-référence, tous les concepts de la T-Box sont mis sous forme normale négative. Ceci est particulièrement visible dans des expressions telles que ***and(not(some(aEcrit, livre)))***, qui ont été transformées en ***and(all(aEcrit,not(livre)))***.

# **Implémentation de la deuxième partie**

## **Saisie de l'Instance et du Concept (Proposition de Type 1)**

L'utilisateur commence par choisir le type de proposition (1) et est ensuite invité à entrer l'identifiant d'une instance. Le prédicat **entrerInstance/1** gère cette étape, vérifiant la validité de l'instance saisie. En cas d'erreur, un message d'erreur est affiché, et l'utilisateur est invité à réessayer.

Après la saisie réussie de l'instance, l'utilisateur est ensuite invité à entrer l'expression d'un concept associé à cette instance. Le prédicat **entrerConcept/1** prend en charge cette étape, vérifiant la validité de l'expression du concept. Si une erreur est détectée, un message d'erreur est affiché, et l'utilisateur est invité à corriger son entrée.

Une fois l'identifiant de l'instance et l'expression du concept validés, la proposition est considérée comme correcte et ajoutée avec succès à la T-Box. Un message de confirmation est affiché, informant l'utilisateur que la proposition a été ajoutée avec succès.

Voici un exemple de l’exécution de **acquisition\_prop\_type1/3** sur la base fournie :

