Université Sorbonne Sciences Jussieu

master 1 : androide et stl
UE LRC

Rapport du projet de Prolog

Élèves :
Loona MACABRE
Carla GIULIANI

Enseignant :
Colette Faucher

15 décembre 2022

Table des matières

1 Première partie

1.1 concept

Le prédicat récursif **concept** prend :
— un argument C pour vérifier si C est concept ;
— deux arguments T et Q pour vérifier si T est bien un identificateur de concept (dans le cas de la Tbox) ou d'instance (dans le cas de la Abox) et Q un concept (on appelle alors concept sur Q) ;
— et trois arguments T, U, V pour contrôler les assertions de rôles et vérifier que T et U sont bien des identificateurs d'instances et V un rôle.

1.2 autoref

Le prédicat récursif **autoref** permet de tester si un concept C est auto-référent en cherchant dans sa définition si il fait appel à un autre concept qui est défini en fonction de C.

1.3 remplace

Le prédicat récursif **remplace** prend en entrée la définition d'un concept C et une nouvelle définition C' obtenue après avoir exprimé tous les concepts non atomiques avec des concepts atomiques.

1.4 traitement-Tbox

Le prédicat récursif **traitement-Tbox** prend la Tbox d'origine T et une nouvelle Tbox T' en entrée et parcourt T en appelant **autoref**, **concept**, **remplace** et **nnf** (fourni) sur chacun de ses éléments pour obtenir T'.

1.5 traitement-Abox

Le prédicat récursif **traitement-Abox** parcourt les assertions de concepts AboxC et de rôles AboxR en appelant **autoref**, **concept**, **remplace** et **nnf** (fourni) sur chacun de leurs éléments pour obtenir respectivement AboxC' et AboxR'.

2 Deuxième partie

2.1 acquisition-prop-type1

Le prédicat **acquisition-prop-type1** prend en entrées l'Abox initiale Abi, l'Abox mise à jour Abe et la Tbox (on n'utilise cependant pas ce dernier argument). Le prédicat demande à l'utilisateur d'écrire la proposition à démontrer puis applique **remplace** et **nnf** sur sa négation qu'il ajoute ensuite à l'Abox pour obtenir Abe.

2.2 acquisition-prop-type2

Le prédicat **acquisition-prop-type2** prend les mêmes arguments que **acquisition-prop-type1**. Il génère une nouvelle instance I, puis demande d'écrire les deux concepts dont on veut prouver l'intersection vide. Ensuite, il applique **remplace** et **nnf** avant d'ajouter le tout à la nouvelle Abox.

3 Troisième partie

3.1 tri-Abox

Le prédicat récursif **tri-Abox** prend en entrées l'ABox étendue (c'est-à-dire l'ABox mise à jour après la deuxième partie) et des listes vides (qui vont se remplir) pour chaque type d'assertions de concept. Le prédicat boucle sur chaque assertion de l'Abox et l'ajoute à la liste correspondante.

3.2 resolution

Le prédicat resolution appelle les prédicats **complete-some**, **transformation-and**, **deduction-all** et **transformation-or**, et représente l'arbre de résolution. Si il termine son exécution sans être interrompu par un clash (i.e en passant par tous les noeuds jusqu'à ce qu'il ne reste plus d'assertions), il termine le programme en affichant un message d'échec.

3.3 complete-some

Le prédicat **complete-some** correspond au noeud Existe dans l'arbre de résolution : il appelle **evolue** pour appliquer la règle $\exists$ sur l'assertion de concept en argument, affiche l'évolution de l'Abox et retourne dans **resolution**.

3.4 transformation-and

Idem en appliquant la règle $\sqcap$.

3.5 deduction-all

Idem en appliquant la règle $\forall$.

3.6 transformation-or

Idem en appliquant la règle $\sqcup$.

3.7 clash

Ce prédicat prend en arguments l'assertion de concept à ajouter ainsi que la liste concernée et permet d'interrompre la résolution en cours lorsqu'il détecte un clash avec un message de succès.

3.8 evolue

evolue prend en entrées l'assertion courante A, les listes Lie,...,Ls et Lie1,...,Ls1 leurs mises à jour respectives. C'est ce prédicat qui s'occupe d'insérer A dans la liste qui lui correspond, après avoir **clash**.

3.9 lisibilite

Le prédicat **lisibilité** permet de transformer une notation préfixe en notation infixe avec les symboles $\exists, \forall, \sqcap$ et $\sqcup$ pour faciliter la lecture. On a aussi écrit les prédicats **lisibiliteListe** et **lisibilitéAbr** qui applique **lisibilite** sur chaque élément de ces structures de données afin d'avoir un affichage agréable dans la console.

3.10 affiche-evolution-ABox

Ce prédicat affiche l'évolution de l'Abox avant d'appliquer une règle et après en utilisant le prédicat **lisibilite**.