Subsomptions en Prolog

Introduction

Le but de ce TME est de programmer en Prolog un algorithme permettant d'inférer des subsomptions pour la logique de description $\mathcal{FL}^-$. On procède de façon progressive en ajoutant des règles pour gérer des cas de plus en plus complexes. On s'assure toujours de faire des inférences correctes : si on trouve que C subsume D selon nos règles, on a la garantie que C subsume bien D. L'inverse n'est pas forcément vrai, le programme, construit progressivement, pouvant être incomplet.

Représentation

Exercice 1 Représentation préfixe en *Prolog*

On traduit les différents opérateurs de concepts et éléments des T-Box et A-Box de la façon suivante :

	$\mathcal{FL}^-$	*Prolog*
Intersection	$C \sqcap D$	`and(C, D)`
Quantification existentielle	$\exists R$	`some(R)`
Quantification universelle	$\forall R.C$	`all(R,C)`
Subsomption	${}^T\boxed{C \sqsubseteq D}$	`subs(C,D)`
Equivalence	${}^T\boxed{C \equiv D}$	`equiv(C,D)`

On travaille dans ce TME sur la T-Box correspondant aux connaissances de l'exercice de TD sur les animaux (TD 4 Exercice 2). Leur traduction en *Prolog* est donnée dans le fichier `LRC_donneesTD4ex2.pl`.

Ouvrir ce fichier pour inspecter les données. Traduire dans les notations classiques de $\mathcal{FL}^-$ les lignes 3, 14, 16, 19, 21 et 22. Identifier les phrases de l'exercice 2 du TD 4 qu'elles traduisent.
Note : $\bot$ n'existant pas dans $\mathcal{FL}^-$, on traite ici `nothing` comme un concept atomique.

Subsomption structurelle pour $\mathcal{FL}^-$

Le but de cette section est d'écrire un ensemble de règles permettant de répondre à des questions du type "est-ce qu'on peut prouver que C est subsumé par D ?" (soit "est-ce que $C \sqsubseteq_s D$?")

Exercice 2 Concepts atomiques

On suppose dans un premier temps que la T-Box ne contient que des expressions du type $A \sqsubseteq B$ où A et B sont des concepts atomiques et que C et D sont aussi atomiques. Pour vérifier si $C \sqsubseteq_s D$ dans ce contexte, on propose les règles suivantes, à écrire dans un fichier `LRC_tme5.pl` dans lequel on aura préalablement copié-collé le contenu du fichier `LRC_donneesTD4ex2.pl`.

```
subsS1(C,C).
subsS1(C,D):-subs(C,D),C\==D.
subsS1(C,D):-subs(C,E),subsS1(E,D).
```

1. Que traduisent ces règles ? Testez les sur les requêtes $canari \sqsubseteq_s animal$, $chat \sqsubseteq_s etreVivant$.
2*. Tester la requête $chien \sqsubseteq_s souris$. Quel problème pose cette requête ? Utiliser la trace pour en identifier la cause.

Pour corriger ce problème, on introduit un troisième argument contenant la liste des subsomptions déjà faites dans une branche. On définit `subsS(C,D)`, qui doit être vérifié quand $C \sqsubseteq_s D$ avec le prédicat auxiliaire `subsS(C,D,L)` où L contient la liste des concepts utilisés dans la preuve de la subsomption. Pour éviter des preuves infinies, on s'interdit de réutiliser un concept déjà présent dans L.

On réécrit donc les règles sur `subsS1(C,D)` pour définir `subsS(C,D,L)`, en rajoutant avant tout appel récursif $E \sqsubseteq_s D$ la condition que E ne soit pas dans L et en ajoutant E à L dans l'appel récursif :

```
subsS(C,D) :- subsS(C,D,[C]).
subsS(C,C,_).
subsS(C,D,_):-subs(C,D),C\==D.
subsS(C,D,L):-subs(C,E),not(member(E,L)),subsS(E,D,[E|L]),E\==D.
```

3. Tester ces nouvelles règles avec *chat* $\sqsubseteq_s$ *etreVivant*, *chien* $\sqsubseteq_s$ *canide*, et *chien* $\sqsubseteq_s$ *chien* et vérifier que le résultat est conforme aux attentes. Tester ensuite la requête *chien* $\sqsubseteq_s$ *souris* qui doit échouer. Inspecter la trace de cette requête.
4. Tester la requête *souris* $\sqsubseteq_s \exists$*mange*. Pourquoi cette requête réussit-elle bien que $\exists$*mange* ne soit pas un concept atomique ?

5*. Que devraient renvoyer les requêtes *chat* $\sqsubseteq_s X$ et $X \sqsubseteq_s$ *mammifere* ? Vérifier que c'est bien le cas (il n'est pas nécessaire d'éliminer les réponses doubles ou le false final).

On suppose maintenant que la T-Box peut aussi contenir des expressions du type $A \equiv B$.

6*. Ecrire des règles permettant de dériver $A \sqsubseteq B$ et $B \sqsubseteq A$ à partir de $A \equiv B$. Tester avant et après ajout des règles la requête *lion* $\sqsubseteq_s \forall$*mange.animal*.

Exercice 3 Gestion de l'intersection

On s'intéresse maintenant aux requêtes contenant des intersections de concepts. On étend donc la définition du prédicat `subsS` avec les règles données dans le fichier `LRC_reglesInter.pl`. Il est important de travailler dans le même fichier et de garder le même nom de prédicat (`subsS`) (afin que la définition soit étendue et non remplacée). Il faut donc recopier les règles du fichier `LRC_reglesInter.pl` dans le fichier `LRC_tme5.pl`, à la suite des règles écrites pour l'exercice précédent.

1. Tester cet ensemble de règles avec les requêtes *chihuahua* $\sqsubseteq_s$ *mammifere* $\sqcap \exists aMaitre$, *chien* $\sqcap \exists aMaitre \sqsubseteq_s$ *pet* et *chihuahua* $\sqsubseteq_s$ *pet* $\sqcap$ *chien*.

2*. Pour chacune des règles, indiquer la situation traitée par la règle et donner un exemple de requête qui ne marcherait pas sans la règle. Ces exemples peuvent utiliser la T-Box donnée directement (pas forcément possible pour toutes les règles), la compléter pour illustrer une situation particulière, ou en proposer une nouvelle assez simple pour illustrer l'utilité de chaque règle.

Exercice 4 Gestion des rôles

On s'intéresse maintenant aux requêtes contenant des rôles qualifiés ($\forall R.C$).

1*. Ecrire une règle permettant de répondre à une requête du type $\forall R.C \sqsubseteq_s \forall R.D$, où C et D sont des concepts à partir de C et D. Il s'agit là encore d'étendre la définition de `subsS` et non de créer un nouveau prédicat.

1. Tester les requêtes *lion* $\sqsubseteq_s \forall$*mange.etreVivant* et $\forall$*mange.canari* $\sqsubseteq_s$ *carnivoreExc*.
2. Tester les requêtes *carnivoreExc* $\sqcap$ *herbivoreExc* $\sqsubseteq_s \forall$*mange.nothing* et (*carnivoreExc* $\sqcap$ *herbivoreExc*) $\sqcap$ *animal* $\sqsubseteq_s$ *nothing*.
   Si besoin, ajouter des règles pour qu'elles réussissent.
   Qu'en est-il de la requête (*carnivoreExc* $\sqcap$ *animal*) $\sqcap$ *herbivoreExc* $\sqsubseteq_s$ *nothing* ? Pourquoi ? (On n'exige pas de modifier le programme pour que cette dernière requête réussisse).
3. Est-il nécessaire d'écrire des règles similaires pour les concepts de la forme $\exists R$?
4. Que devraient renvoyer les requêtes *lion* $\sqsubseteq_s X$ et $X \sqsubseteq_s$ *predateur* ? Voir si c'est bien le cas avec vos règles (il n'est pas nécessaire d'éliminer les réponses doubles ou le false final).

Exercice 5 Complétude

Un ensemble de règles ainsi produit est complet pour un langage donné s'il peut prouver toute subsomption $C \sqsubseteq D$ correcte à partir du moment où tous les termes de la T-Box et de la requête appartiennent au langage donné.

L'ensemble de règles écrites est-il complet pour $\mathcal{FL}^-$?