TP numéro 4

Intelligence artificielle, ENSIIE

Semestre 4, 2022–23

Exercice 1 : Coupures

1. Définir trois versions du prédicat `retire(L1,L2,L3)` qui, étant données les listes `L1` et `L2`, construit la liste `L3` qui contient les éléments de `L1` qui n'appartiennent pas à `L2` :
   - avec un test ;
   - avec une coupure ;
   - en enlevant la coupure.

   Comparer le fonctionnement de ces trois versions.
2. Définir le prédicat `insere(X,L,R)` qui, en supposant que la liste `L` est triée, insère `X` dans `L` pour obtenir la liste `R` triée.
   Donner un version sans coupure et une avec.
3. Tester ces deux versions sur des exemples complets.
   (Par exemple `insere(5,[1,3,7,9],[1,3,5,7,9])`)
4. Tester ces deux versions en mode génération.
   (Par exemple `insere(5,[1,3,7,9],R)` ou `insere(5,L,[1,3,5,7,9])`)
5. Définir un prédicat `trie(L,R)` qui trie `L` en `R` en utilisant un tri par insertion. Quelle version de `insere` vaut-il mieux utiliser ?

Exercice 2 : Crible d'Ératosthène

On cherche à générer une liste de nombres premiers en utilisant la méthode d'Ératosthène.

1. Définir un prédicat `multiple(N, M)` qui est vrai quand `M` est un multiple de `N`. On pourra utiliser l'opérateur `mod` qui calcule le reste de la division euclidienne.
2. Définir un prédicat `from_to(B, E, L)` qui est vrai quand `L` contient la liste des entiers compris entre `B` et `E` inclus.
3. Définir un prédicat `clean_multiple(N, L, R)` qui est vrai quand `R` est la liste `L` de laquelle on a retiré tous les multiples de `N`. Il peut être opportun d'utiliser une coupure.
4. Définir un prédicat `erasto(L, R)` qui applique la méthode d'Ératosthène à `L` pour obtenir `R` : les multiples du premier élément N de `L` sont retirés de la suite de `L`, puis on recommence récursivement avec la liste obtenue et on remet N en tête.

5. Définir un prédicat `primes_up_to(N, L)` qui est vrai quand `L` est la liste des nombres premiers jusqu'à `N`. Pour cela, il faut générer la liste de tous les entiers de 2 à `N`, puis appliquer la méthode d'Ératosthène dessus.

Exercice 3 : Négation et pour tout

6. Définir un prédicat `disjoint(L,R)` qui est vrai si `L` et `R` n'ont pas d'éléments en commun. On pourra utiliser le prédicat `member(X,L)` et une négation.
7. Donner une autre définition de `disjoint` en utilisant `forall`.