TP3 : Résolution - Récursivité

1 Généalogie

Le but des questions qui suivent est de constater que bien que les programmes en PROLOG traduisent des formules de la logique des prédicats, des formulations différentes mais logiquement équivalentes conduisent à des programmes dont le comportement opérationnel diffère.

Vous allez le voir en vous appuyant sur quatre variantes du même prédicat `descendant_de/2` défini logiquement par

$$\forall X \forall Y \text{ descendant\_de}(X,Y) \Longleftrightarrow (\text{enfant\_de(X,Y)}) \vee (\exists Z(\text{enfant\_de(X,Z)} \wedge \text{descendant\_de(Z,Y)})),$$

où `enfant_de/2` est un prédicat définissant une relation de filiation : `enfant_de(X,Y)` signifie `X` est un enfant de `Y`.

Question 1

Q 1–1 Récupérez le fichier `genealogie1.pl`, et étudiez les régles définissant le prédicat `descendant_de`.

Q 1–2 Chargez ce fichier dans un interpréteur PROLOG. Passez en mode trace (prédicat **`trace/0`**), et soumettez le but `descendant_de(anne,emily)`. Comptez combien d'étapes sont franchies par PROLOG pour parvenir à une réponse.

Q 1–3 Quittez le mode trace (prédicat `notrace`). Soumettez le bur `descendant_de(X,Y)`. Combien de solutions obtenez-vous ?

Question 2 Même question avec le fichier `genealogie2.pl`.

Question 3 Même question avec le fichier `genealogie3.pl`.

Question 4 Même question avec le fichier `genealogie4.pl`.

Question 5 Quelle est à votre avis la meilleure façon d'exprimer en PROLOG le prédicat `descendant_de` ? Pourquoi ?

2 Entiers de Peano

Le but de cette partie est de manipuler des entiers dans la représentation de Peano [1]. Bien entendu, PROLOG comme tout langage de programmation peut travailler avec des nombres entiers et possède des opérateurs arithmétiques et des opérateurs de comparaison.

En suivant Peano, on peut définir tous les nombres entiers à partir du premier d'entre eux : 0, et de la fonction successeur qui à un entier n associe l'entier $n+1$. Ainsi l'entier 4 peut être décrit comme

$$4 = ((((0+1)+1)+1)+1),$$

ou encore, en notant `succ` cette fonction successeur,

$$4 = \text{succ}(\text{succ}(\text{succ}(\text{succ}(0)))).$$

Il s'agit ici de travailler sur une représentation abstraite des entiers vus comme des termes construits à partir d'une constante (`zero`) et d'un symbole de fonction (`succ`). En voici une grammaire :

$$\textit{Entier} \quad ::= \quad \text{zero} \mid \text{succ}(\textit{Entier}).$$

La table 1 donne quelques exemples d'entiers dans cette représentation.

Question 6

Q 6–1 Réalisez le prédicat `entier/1` qui est satisfait lorsque son paramètre est un terme représentant un entier de Peano, et ne l'est pas pour tout autre terme.

Pour cela il vous suffit de suivre au plus près la grammaire qui définit les entiers de Peano.

Q 6–2 Testez votre prédicat pour les buts suivants

1. `entier(zero)`
2. `entier(succ(succ(zero)))`

1. Giuseppe Peano, 1858-1932, mathématicien italien qui s'est intéressé, entre autres choses, aux fondements de l'arithmétique et en a proposé une axiomatisation.

n	Représentation de Peano
0	`zero`
1	`succ(zero)`
2	`succ(succ(zero))`
3	`succ(succ(succ(zero)))`
4	`succ(succ(succ(succ(zero))))`

TABLE 1 – Quelques entiers en représentation de Peano

3. `entier(succ(succ(0)))`
4. `entier(f(zero))`
5. `entier(X)`

Pour ce dernier but, combien y a-t-il de solutions ?

Question 7

Q 7–1 Réalisez un prédicat `inf_ou_egal/2` satisfait lorsque le premier terme est un entier de Peano inférieur ou égal au second terme.

Q 7–2 Testez votre prédicat sur les buts suivants et donnez le cas échéant le nombre de solutions.

1. `inf_ou_egal(succ(zero),succ(succ(zero)))`
2. `inf_ou_egal(succ(succ(zero)), succ(zero))`
3. `inf_ou_egal(X,succ(succ(zero)))`
4. `inf_ou_egal(succ(zero),Z)`

Comment interpréter la réponse fournie pour le dernier but ?

Question 8 Il s'agit maintenant de réaliser un prédicat `add` pour l'addition de deux entiers de Peano. Ce prédicat est d'arité 3, et il est satisfait si et seulement si les deux premiers termes sont des entiers de Peano quelconques, et le troisième l'entier de Peano représentant la somme des deux premiers.

```
?- add(succ(zero),succ(zero),succ(succ(zero))).
true.

?- add(succ(zero),succ(zero),succ(zero)).
false.

?- add(succ(zero), succ(zero), R).
R = succ(succ(zero)) ;
false.
```

Q 8–1 En prenant en compte les égalités arithmétiques

$$n + 0 = n$$
$$n + (p + 1) = (n + p) + 1$$

et en notant que l'opération $+1$ sur un entier de Peano consiste à ajouter un `succ` sur cet entier, réalisez le prédicat `add/3`.

Q 8–2 Testez votre prédicat sur les buts en donnant le cas échéant le nombre de solutions.

1. `add(succ(zero), succ(succ(zero)), succ(succ(zero)))`
2. `add(succ(zero), succ(succ(zero)), succ(succ(succ(zero))))`
3. `add(succ(zero), succ(succ(zero)), R)`
4. `add(succ(zero), Z, succ(succ(succ(zero))))`
5. `add(succ(zero), X, Y)`

Q 8–3 En déduire un prédicat `sub/3` pour la soustraction.

Question 9 Réalisez un prédicat `mult/3` pour la multiplication.