Université Paris Diderot - L3 Année 2018-2019

TD de Logique n° 1

Raisonnement par induction

Exercice 1 [Grammaire] On considère trois constantes $\mathtt{a}$, $\mathtt{c}$, $\mathtt{d}$. On définit l'ensemble E comme le plus petit ensemble des expressions construites sur l'alphabet $\{\mathtt{a}, \mathtt{c}, \mathtt{d}\}$ vérifiant les conditions suivantes :

$$\frac{}{\mathtt{a} \in E} \qquad \frac{e_1 \in E \quad e_2 \in E}{\mathtt{c}(e_1, e_2) \in E} \qquad \frac{e_1 \in E \quad e_2 \in E \quad e_3 \in E}{\mathtt{d}(e_1, e_2, e_3) \in E}$$

Le nombre de symboles $\mathtt{a}$, $\mathtt{c}$ et $\mathtt{d}$ dans une expression $e \in E$ est noté $|e|_{\mathtt{a}}$, $|e|_{\mathtt{c}}$ et $|e|_{\mathtt{d}}$ respectivement.

1. Munir l'ensemble E d'un ordre bien fondé. Quel est son élément minimal ?
2. Montrer par induction structurelle la propriété suivante :

$$\text{Pour toute expression } e \in E,\ |e|_{\mathtt{a}} = 2 * |e|_{\mathtt{d}} + |e|_{\mathtt{c}} + 1.$$

3. Proposer une démonstration de la même propriété par induction complète (sur les entiers).

Exercice 2 [Arbres binaires]

1. Rappeler la définition inductive de l'ensemble des arbres binaires ainsi que la définition de l'ordre bien fondé sur cet ensemble.
2. Définir la fonction h qui associe à chaque arbre binaire sa hauteur (entier naturel), sachant que la hauteur de l'arbre vide est 0.
3. Définir la fonction $|\cdot|$ qui associe à chaque arbre binaire le nombre de ses noeuds (ou de ses sous-arbres non vides).
4. Un arbre binaire t est *parfait* si, pour tout sous-arbre de t de la forme $\mathtt{ab}(A_1, A_2)$, on a $h(A_1) = h(A_2)$. Donner une définition inductive de l'ensemble des arbres binaires parfaits.
5. Prouver par induction structurelle que pour tout arbre binaire parfait t, on a

$$|t| = 2^{h(t)} - 1$$

Exercice 3 [Terminaison et ordre lexicographique] On considère la fonction `caml` suivante

```
let rec f c = match c with
   (0, p) -> 42
  |(1, p) when p>=7 -> f(1, p-7)
  |(n, p) -> f(n-1, f(n-1, p+7));;
```

On muni $\mathbb{N}^2$ de l'ordre lexicographique défini comme suit :

$$(m, q) >_{\text{lex}} (n, p) \text{ lorsque } (m > n) \text{ ou } (n = m \text{ et } q > p).$$

1. Donner un argument pour montrer que l'ordre $>_{\text{lex}}$ est bien fondé. Déterminer le ou les éléments minimaux de cet ordre.

2. Quel est le résultat de chaque appel de `f` (pour tout couple $(n, p) \in \mathbb{N}^2$ en entrée) ?
3. Montrer la propriété précédente.

Exercice 4 [Induction mutuelle] On considère les forêts et les arbres définis en cours. Le but de cet exercice est de démontrer la proposition suivante : « Pour tout arbre dont les nœuds sont des entiers non nuls, le nombre de nœuds est inférieur ou égal à la somme des valeurs des nœuds de l'arbre ».

1. Rappeler la définition mutuellement inductive des forêts et des arbres.
2. Définir une fonction `count` qui compte le nombre de nœuds d'un arbre (on utilisera une fonction intermédiaire `count_foret` qui compte le nombre de nœuds d'une forêt).
3. Définir une fonction `sum` qui somme les valeurs des nœuds d'un arbre (on utilisera une fonction intermédiaire `sum_foret` qui somme les valeurs des nœuds d'une forêt).
4. Montrer par *induction structurelle* sur les ensembles des arbres et des forêts la proposition suivante :

   « Pour tout arbre A et pour toute forêt F, si la valeur de chacun des nœuds est plus grande que 1, alors $\texttt{count}\ A \leq \texttt{sum}\ A$ et $\texttt{count\_foret}\ F \leq \texttt{sum\_foret}\ F$ »

Exercice 5 [Arbres binaires (suite)]
On considère une fonction `f` définie sur les couples d'arbres binaires comme suit :

$$\begin{array}{lcl} \texttt{f}(\texttt{vide}, A_1) & := & A_1 \\ \texttt{f}(\texttt{ab}(A_1, A_2), A_3) & := & \texttt{f}(A_1, \texttt{ab}(A_3, A_2)) \end{array}$$

1. Soit A l'arbre suivant (dont les nœuds ont été numérotés pour plus de clarté et `v` signifie `vide`) :
   $$A = n_1(n_2(n_4(\texttt{v}, \texttt{v}), n_5(n_6(\texttt{v}, \texttt{v}), \texttt{v})), n_3(\texttt{v}, \texttt{v}))$$
   Dessiner l'arbre A en entier (avec tous ses nœuds `vide`), puis calculer $\texttt{f}(A, \texttt{v})$.
2. Montrer que le calcul de la fonction `f` termine quels que soient les arbres fournis en argument.
3. Soit $B = \texttt{f}(A, \texttt{v})$. Calculer $\texttt{f}(B, \texttt{v})$. Que remarquez-vous ?
4. Montrer que pour tout arbre A, $\texttt{f}(\texttt{f}(A, \texttt{v}), \texttt{v}) = A$.
   Indication : commencer par montrer (par induction) la propriété suivante :
   $$\text{pour tout arbre } A, \text{ pour tout arbre } B, \quad \texttt{f}(\texttt{f}(A, B), \texttt{v}) = \texttt{f}(B, A).$$

Exercice 6 [Mots bien parenthésés] Dans cet exercice, on travaille avec un alphabet Σ dans lequel on distingue deux lettres particulières $\texttt{[} \in \Sigma$ (« crochet ouvrant ») et $\texttt{]} \in \Sigma$ (« crochet fermant »), les autres éléments de Σ étant arbitraires. On note Σ^* l'ensemble de tous les mots sur l'alphabet Σ et E l'ensemble des mots *bien parenthésés*, qui est défini inductivement à partir des règles suivantes (ε désigne le mot vide) :

$$\frac{}{\varepsilon \in E} \qquad \frac{x \in \Sigma \quad x \neq \texttt{[} \quad x \neq \texttt{]}}{x \in E} \qquad \frac{u \in E}{\texttt{[}u\texttt{]} \in E} \qquad \frac{u \in E \quad v \in E}{uv \in E}$$

À toute lettre $x \in \Sigma$, on associe un *poids* $p(x) \in \mathbb{N}$ défini par :

$$p(\texttt{[}) = 1, \qquad p(\texttt{]}) = -1, \qquad p(x) = 0 \quad \text{si } (x \neq \texttt{[} \text{ et } x \neq \texttt{]})$$

On étend cette fonction de poids à tous les mots sur l'alphabet Σ en posant

$$p(x_1x_2\cdots x_n) \;=\; p(x_1)+p(x_2)+\cdots+p(x_n)$$

pour tout mot $u = x_1x_2\cdots x_n \in \Sigma^*$. En particulier, pour $n = 0$, on a $p(\varepsilon) = 0$.

1. Montrer que tout mot bien parenthésé $u \in E$ satisfait les deux propriétés suivantes :

   (P-a) $p(u) = 0$.

   (P-b) Si v est un préfixe de u, alors $p(v) \geq 0$.

2. Montrer réciproquement que si $u \in \Sigma^*$ satisfait les propriétés (P-a) et (P-b), alors u est bien parenthésé.

3. En déduire un algorithme qui permet de tester si un mot est bien parenthésé.