LIFLC – Logique classique
TD2 – Induction

Licence informatique UCBL – Automne 2018–2019

Les (parties d') exercices noté(e)s avec † sont plus difficiles.

Exercice 1 : Ensembles inductifs

Soit E et F deux ensembles. Expliquer quels sont les ensembles inductifs définis à partir des 3 ensembles de règles suivants. Expliquer les différences entre ces trois ensembles.

1. Ensemble N
   - $leaf_N \in N$
   - si $n_1 \in N$, $n_2 \in N$ et si $e \in E$ alors $node_N(n_1, e, n_2) \in N$
2. Ensemble L
   - si $e \in E$, alors $leaf_L(e) \in L$
   - si $n_1 \in L$ et $n_2 \in L$ alors $node_L(n_1, n_2) \in L$
3. Ensemble LH
   - si $e \in E$ alors $leaf_{LH}(e) \in LH$
   - si $n_1 \in LH$, $n_2 \in LH$ et $f \in F$, alors $node_{LH}(n_1, f, n_2) \in LH$

Exercice 2 : Ensemble inductif des arbres binaires de recherche

Soit E un ensemble muni d'un ordre total $\leq_E$.

1. Donner une définition par induction de l'ensemble Bin_E des arbres binaires contenant des éléments de E. On souhaite avoir une représentation d'arbre vide dans Bin_E.
2. Définir la fonction récursive *elements* qui renvoie l'ensemble des éléments de E contenus dans un arbre binaire. On commencera par donner la signature (domaine et co-domaine) de cette fonction.
3. Montrer par induction sur Bin_E que pour un arbre binaire a comportant n occurrences de l'arbre vide, $\mid elements(a) \mid \leq n - 1$.
4. Donner une définition par induction de l'ensemble $BinRech_E$ des arbres binaires de recherche contenant des éléments de E.
5. Définir la fonction récursive *plusPetitElement* qui renvoie le plus petit élément d'un arbre binaire de recherche [1]. Dans le cas où un tel élément n'existe pas, la fonction reverra la valeur spéciale $\Diamond$. On commencera par donner la signature de cette fonction.
6. Montrer que la fonction *plusPetitElement* est correcte. Pour cela, montrer par induction sur Bin_E que : soit $plusPetitElement(a) = min(elements(a))$, soit $elements(a) = \emptyset$ et $plusPetitElement(a) = \Diamond$.

1. on souhaite ici que la complexité de la fonction soit linéaire dans la hauteur de l'arbre

Exercice 3 : Plus petit ensemble stable par des règles de constructions †

Soit E un ensemble. On considère les deux règles de construction de listes suivantes :

— [] est une liste (on note cette règle $\mathcal{R}_1$)

— si l' est une liste et si $e \in E$, alors $cons(e, l')$ est une liste (on note cette règle $\mathcal{R}_2$)

Soit F un ensemble tel qu'il admet au moins un sous-ensemble G stable par ces règles de construction (*i.e.* $[] \in G$ et si $e \in E$ et $l' \in G$, alors $cons(e, l') \in G$). Montrer qu'il existe un unique plus petit (au sens de l'inclusion) sous-ensemble de F stable par ces règles de construction.

Indice : considérer l'intersection de tous les sous-ensembles de F stables par ces règles de construction.

Exercice 4 : Ordre bien fondé sur un ensemble inductif †

Soit E un ensemble. On considère l'ensemble inductif des listes $List_E$ construit à partir des règles suivantes :

— [] est une liste

— si l' est une liste et si $e \in E$, alors $cons(e, l')$ est une liste

Soit la relation binaire $\lhd$ définie sur $List_E \times List_E$ par $l' \lhd l$ si et seulement si on peut trouver $e \in E$ tel que $l = cons(e, l')$. Soit $\leq$ la fermeture réflexive transitive de $\lhd$. On suppose que :

a. *cons* est injective ;

b. qu'il n'existe pas de liste l et d'élément e tels que $[] = cons(e, l)$.

Montrer que $\leq$ est un ordre bien fondé sur $List_E$. On admettra que $\leq$ est un ordre.

Indice : On pourra montrer par induction que pour toute liste l, il n'existe pas de suite infinie strictement décroissante commençant par une liste $l' \leq l$.

Corrections

Solution de l'exercice 1

L'objectif de l'exercice est de faire travailler les étudiants afin qu'ils se construisent une intuition de ces structures. On peut donner des exemples, dessiner les arbres au tableau, etc. Quelques exemples de remarques/différences :

- Les trois ensembles sont des variations d'arbres binaires.
- Seul N permet d'avoir un arbre vide $leaf_N$.
- N stocke les valeurs dans des nœuds internes, L dans les feuilles et LH dans les deux, mais les valeurs sont issues d'ensembles différents.
- En prenant LH avec E singleton, on peut encoder N sur les éléments de F (*i.e.* on peut construire un isomorphisme entre $LH(E, F)$ et $N(F)$).
- Comme exemple d'application de $LH(E, F)$, on peut prendre $E = \mathcal{N}$ et $F = \{+, \times\}$ pour représenter des expressions arithmétiques.

Solution de l'exercice 2

1. c'est l'ensemble N de l'exercice 1. L'ensemble Bin_E est le plus petit ensemble tel que :
   - $\square \in Bin_E$
   - Si $e \in E$, $a_1 \in Bin_E$ et $a_2 \in Bin_E$ alors $node(a_1, e, a_2) \in Bin_E$.
2. $$elements : Bin_E \to \mathcal{P}(E)$$
   $$elements(a) = \begin{cases} \emptyset & \text{si } a = \square \\ \{e\} \cup elements(a_1) \cup elements(a_2) & \text{si } a = node(a_1, e, a_2) \end{cases}$$
   Plusieurs erreurs sur le type à anticiper, comme $elements : Bin_E \to (E)$.
3. On montre que la propriété est stable par les règles de construction de Bin_E. On procède par cas sur la règle de construction utilisée :
   - cas $a = \square$:
     $n = 1$ et $elements(a) = 0$, donc l'égalité est vérifiée.
   - cas $a = node(a_1, e, a_2)$:
     Soit n_1 (resp. n_2) le nombre d'occurrences de $\square$ dans a_1 (resp. a_2). On rappelle les hypothèses d'induction : $|elements(a_1)| \leq n_1 - 1$ et $|elements(a_2)| \leq n_2 - 1$.
     Comme $elements(a) = \{e\} \cup elements(a_1) \cup elements(a_2)$, on a $|elements(a)| \leq 1 + |elements(a_1)| + |elements(a_2)|$.
     On en déduit $|elements(a)| \leq 1 + n_1 - 1 + n_2 - 1$.
     Or $n = n_1 + n_2$, donc $|elements(a)| \leq n - 1$.

   Comme Bin_E est *le* plus petit ensemble stable par ses propres règles de construction, il est forcément inclus dans l'ensemble des arbres binaires vérifiant la propriété. On vient en effet de montrer que ce $2^{\text{ième}}$ ensemble est lui aussi stable par ces règles.
4. L'ensemble $BinRech_E$ est le plus petit ensemble tel que :
   - $\square \in BinRech_E$
   - Si les conditions suivantes sont toutes vérifiées
     - $e \in E$,
     - $a_1 \in BinRech_E$,
     - $a_2 \in BinRech_E$,

— soit $e \geq_E max(elements(a_1))$, soit $a_1 = \square$,

— soit $e \leq_E min(elements(a_2))$ soit $a_2 = \square$,

alors $node(a_1, e, a_2) \in BinRech_E$.

Remarque : $BinRech_E \subseteq Bin_E$ car Bin_E est stable par ces règles. On utiliser cette remarque pour proposer une définition alternative équivalente de $BinRech_E$:

$$BinRech_E = \{t \in Bin_E \mid \quad \text{si } t = node(a_1, e, a_2) \text{ alors } (e \geq_E max(elements(a_1) \text{ ou } a_1 = \square) \text{ et } (e \leq_E min(elements(a_2) \text{ ou } a_2 = \square)\}$$

5.

$$plusPetitElement : Bin_E \rightarrow E \cup \{\lozenge\}$$

$$plusPetitElement(a) = \begin{cases} \lozenge & \text{si } a = \square \\ e & \text{si } a = node(a_1, e, a_2) \text{ et } a_1 = \square \\ plusPetitElement(a_1) & \text{si } a = node(a_1, e, a_2) \text{ et } a_1 \neq \square \end{cases}$$

6. On montre par induction et par cas sur a que soit $plusPetitElement(a) = min(elements(a))$, soit $elements(a) = \emptyset$ et $plusPetitElement(a) = \lozenge$.

   — Si $a = \square$, alors il suffit de remarquer que, par définition, $elements(\square) = \emptyset$ et $plusPetitElement(\square) = \lozenge$.

   — Supposons $a = node(e, a_1, a_2)$.
   Par induction, soit $plusPetitElement(a_1) = min(elements(a_1))$, soit $elements(a_1) = \emptyset$ et $plusPetitElement(a_1) = \lozenge$.
   Si $elements(a_1) = \emptyset$ alors $a_1 = \square$. Dans ce cas, par définition $e \leq_E min(elements(a_2))$, et on a donc $e = min(elements(a_2) \cup \{e\} \cup \emptyset) = min(elements(a))$.
   Sinon $plusPetitElement(a_1) = min(elements(a_1))$. Par définition $e \leq_E min(elements(a_2))$ et définition $e \geq_E max(elements(a_1) \geq_E min(elements(a_1))$, donc $min(elements(a_1)) = min(elements(a_1) \cup elements(a_2) \cup \{e\}) = min(elements(a))$, et donc $plusPetitElement(a) = plusPetitElement(a_1) = min(elements(a_1)) = min(elements(a))$.

Solution de l'exercice 3

Soit $\mathcal{F}_{stable}$ l'ensemble des sous-ensembles de F qui sont stables par les règles de construction de listes. Soit $H = \bigcap \mathcal{F}_{stable}$ l'intersection de tous les ensembles dans $\mathcal{F}_{stable}$. On peut remarquer que :

— $[] \in H$ car la première règle de construction$\mathcal{R}_1$ impose que tous les éléments de $\mathcal{F}_{stable}$ contiennent [], et donc il en est de même pour leur intersection.

— Soit $l' \in H$ et $e \in E$. Soit $G' \in \mathcal{F}_{stable}$ quelconque. On a $H \subseteq G'$, donc $l' \in G'$. Par $\mathcal{R}_2$, on en déduit que $cons(e, l') \in G'$. Donc $cons(e, l') \in \bigcap \mathcal{F}_{stable}$.

H est donc stable par $\{\mathcal{R}_1, \mathcal{R}_2\}$, donc $H \in \mathcal{F}_{stable}$. Comme c'est l'intersection de tous les éléments de $\mathcal{F}_{stable}$, il est plus petit que tous les autres et il a été défini de manière unique.

Solution de l'exercice 4

Preuve de bonne fondation :

— Par (b.), il n'existe pas de liste $l \lhd []$. Donc il n'existe pas de suite infinie décroissante commençant par [].

— Supposons que $l = cons(e, l')$.
Hypothèse d'induction sur l' : il n'existe pas de suite infinie strictement décroissante commençant par $l'' \leq l'$. Soit une suite infinie décroissante (S_i) commençant par $l'' \leq l$. Comme *cons* est injective, le seul élément l_1 tel que $l_1 \lhd l$ est l'. Or, par ce qui précède, $l'' \not\leq l'$ (cela contredirait l'hypothèse d'induction). Donc $S_0 = l$. Cependant $S_1 \leq S_0 = l$, donc $S_1 \leq l'$. Or (S_{i+1}) devrait être une suite infinie strictement décroissante, ce qui est impossible par l'hypothèse d'induction. Donc il n'existe pas de suite infinie décroissante commençant par $l'' \leq l$.

Pour les curieu.x.ses, voici une preuve que $\leq$ est un ordre :

Transitivité, réflexivité Par définition, $\leq$ est une fermeture transitive réflexive.

Antisymmétrie Soit $l \leq l'$ et $l' \leq l$. On peut avoir $l = l'$. Sinon $l \neq l'$ et :

1. Il existe une suite $l_1, \dots, l_k, \dots, l_m$, telle que $l_i \lhd l_{i+1}$, $l_1 = l_m = i$ et $l_k = l'$.
2. Par (b.), on sait que pour tout $1 \leq i \leq m$, $l_i \neq []$.
3. Comme *cons* est injective, si $l''' \in \{l_1, \dots, l_{m-1}\}$, il existe un unique l'' et un unique e tel que $l'' = cons(e, l'')$. On peut alors remarquer que dans ce cas $l''' \in \{l_1, \dots, l_{m-1}\}$.
4. Considérons l'ensemble $List'_E = List_E \setminus \{l_1, \dots, l_{m-1}\}$.
   (a) $[]$ n'étant pas dans $\{l_1, \dots, l_{m-1}\}$, $[] \in List'_E$.
   (b) Soit $l'' \in List'_E$. Comme $List'_E \subseteq List_E$, on a $l'' \in List_E$ et donc pour tout $e \in E$, $cons(e, l'') \in List_E$. Or par ce qui précède, si $cons(e, l'') \in \{l_1, \dots, l_{m-1}\}$, alors $l'' \in \{l_1, \dots, l_{m-1}\}$, ce qui contredit $l'' \in List'_E$. Donc $cons(e, l'') \in List_E \setminus \{l_1, \dots, l_{m-1}\} = List'_E$.

   Donc $List'_E$ est stable par les règles de construction de liste.
5. Or $List_E$ est le plus petit ensemble stable par ces règles et $List'_E \subset List_E$, ce qui est une contradiction.
6. Donc le cas $l \neq l'$ est impossible.

Donc $\leq$ est bien antisymétrique.