TD8: Induction structurelle
Correction

MP2I Lycée Pierre de Fermat

Exercice 1. *Entiers de Péano*

Question 1. On définit inductivement la fonction **int** comme suit :

— $\mathbf{int}(Z) = 0$

— Pour $a \in \mathcal{P}$, $\mathbf{int}(S(a)) = 1 + \mathbf{int}(a)$

Question 2. Montrons par induction sur la structure de $\mathcal{P}$ la propriété suivante :

$$\forall a \in \mathcal{P}, I(a) : \text{“}\forall b \in \mathcal{P}, \mathbf{int}(\mathbf{add}(a,b)) = \mathbf{int}(a) + \mathbf{int}(b)\text{”}$$

Il y a deux cas :

— $a = Z$: Soit $b \in \mathcal{P}$. $\mathbf{int}(\mathbf{add}(Z,b)) = \mathbf{int}(b)$ par définition de **add**, et $\mathbf{int}(Z) + \mathbf{int}(b) = \mathbf{int}(b)$ par définition de **int**. D'où $I(Z)$.

— $a = S(a')$ avec $a' \in \mathcal{P}$. Hypothèse d'induction : $I(a')$, i.e. $\forall b \in \mathcal{P}, \mathbf{int}(\mathbf{add}(a',b)) = \mathbf{int}(a) + \mathbf{int}(b)$. On a (**Note : pour cette question je mets toutes les justifications pour vous expliquer chaque passage de ligne, mais dans une vraie copie ce n'est pas la peine, en revanche il faut systématiquement le faire pour les hypothèses d'induction et pour l'utilisation de résultats des questions précédents**) :

$$\begin{aligned}
\mathbf{int}(\mathbf{add}(a,b)) &= \mathbf{int}(\mathbf{add}(S(a'),b)) \\
&= \mathbf{int}(S(\mathbf{add}(a',b))) && \text{par définition de } \mathbf{add} \\
&= 1 + \mathbf{int}(\mathbf{add}(a',b)) && \text{par définition de } \mathbf{int} \\
&= 1 + \mathbf{int}(a') + \mathbf{int}(b) && \text{par hypothèse d'induction} \\
&= \mathbf{int}(a) + \mathbf{int}(b) && \text{par définition de } \mathbf{int}
\end{aligned}$$

D'où $I(a)$.

Question 3. On raisonne par induction structurelle sur $a \in \mathcal{P}$. Il y a deux cas :

— $a = Z$: $\mathbf{add}(a,Z) = \mathbf{add}(Z,Z) = Z = a$

— $a = S(a')$ avec $a' \in \mathcal{P}$. Hypothèse d'induction : $\mathbf{add}(a',Z) = a'$. On a :

$$\begin{aligned}
\mathbf{add}(a,Z) &= \mathbf{add}(S(a'),Z) \\
&= S(\mathbf{add}(a',Z)) \\
&= S(a') && \text{par hypothèse d'induction} \\
&= a
\end{aligned}$$

Question 4. On raisonne par induction structurelle sur $a \in \mathcal{P}$. Il y a deux cas :

— $a = Z$:

$$\begin{aligned} \mathbf{add}(S(Z), b) &= S(\mathbf{add}(Z, b)) \\ &= S(b) \\ &= \mathbf{add}(Z, S(b)) \end{aligned}$$

— $a = S(a')$ avec $a' \in \mathcal{P}$. Hypothèse d'induction : $\forall b \in \mathcal{P}, \mathbf{add}(S(a'), b) = S(\mathbf{add}(a', b))$. Soit $b \in \mathcal{P}$. On a :

$$\begin{aligned} \mathbf{add}(S(a), b) &= \mathbf{add}(S(S(a')), b) \\ &= S(\mathbf{add}(S(a'), b)) \\ &= S(\mathbf{add}(a', S(b)) \quad \text{par hypothèse d'induction} \\ &= \mathbf{add}(S(a'), S(b)) \\ &= \mathbf{add}(a, S(b)) \end{aligned}$$

Question 5. Montrons par induction structurelle sur $a \in \mathcal{P}$ la propriété suivante :

$$\forall a \in \mathcal{P}, \mathbf{Com}(a) : \text{“}\forall b \in \mathcal{P}, \mathbf{add}(a, b) = \mathbf{add}(b, a)\text{”}$$

Il y a deux cas :

— $a = Z$. Alors, pour $b \in \mathcal{P}$, $\mathbf{add}(a, b) = \mathbf{add}(Z, b) = b = \mathbf{add}(b, Z)$ d'après la question 3. Donc $\mathbf{Comm}(Z)$.

— $a = S(a')$ avec $a' \in \mathcal{P}$. HI : $\mathbf{Comm}(a')$, i.e. $\forall b \in \mathcal{P}, \mathbf{add}(a', b) = \mathbf{add}(b, a')$. Pour $b \in \mathcal{P}$, on a :

$$\begin{aligned} \mathbf{add}(a, b) &= \mathbf{add}(S(a'), b) \\ &= S(\mathbf{add}(a', b) \\ &= S(\mathbf{add}(b, a') \quad \text{par HI} \\ &= \mathbf{add}(S(b), a') \\ &= \mathbf{add}(b, S(a')) \quad \text{d'après la question 4} \\ &= \mathbf{add}(b, a) \end{aligned}$$

D'où $\mathbf{Com}(a)$.

Ainsi, par principe d'induction structurelle, $\forall a, b \in \mathcal{P}, \mathbf{add}(a, b) = \mathbf{add}(b, a)$.

Question 6. Montrons que $\forall a, b, c \in \mathcal{P}, \mathbf{add}(a, \mathbf{add}(b, c)) = \mathbf{add}(\mathbf{add}(a, b), c)$. Fixons $b, c \in \mathcal{P}$, et montrons par induction structurelle sur $a \in \mathcal{P}$ que :

$$\forall a \in \mathcal{P}, \mathbf{Assoc}(a) : \text{“}\mathbf{add}(a, \mathbf{add}(b, c)) = \mathbf{add}(\mathbf{add}(a, b), c)\text{”}$$

Il y a deux cas :

— $a = Z$. On a :

$$\begin{aligned} \mathbf{add}(a, \mathbf{add}(b, c)) &= \mathbf{add}(Z, \mathbf{add}(b, c)) \\ &= \mathbf{add}(b, c) \\ &= \mathbf{add}(\mathbf{add}(Z, b), c) \\ &= \mathbf{add}(\mathbf{add}(a, b), c) \end{aligned}$$

— $a = S(a')$ avec $a' \in \mathcal{P}$. Hypothèse d'induction : $\mathbf{add}(a', \mathbf{add}(b, c)) = \mathbf{add}(\mathbf{add}(a', b), c)$. On a :

$$
\begin{aligned}
\mathbf{add}(a, \mathbf{add}(b, c)) &= \mathbf{add}(S(a'), \mathbf{add}(b, c)) \\
&= S(\mathbf{add}(a', \mathbf{add}(b, c)) \\
&= S(\mathbf{add}(\mathbf{add}(a', b), c) \quad \text{par HI} \\
&= \mathbf{add}(S(\mathbf{add}(a', b)), c) \\
&= \mathbf{add}(\mathbf{add}(S(a'), b), c) \\
&= \mathbf{add}(\mathbf{add}(a, b), c)
\end{aligned}
$$

Question 7. Il y a plusieurs manières de définir la multiplication, chacune menant à une preuve plus ou moins compliquées et nécessitant plus ou moins de lemmes intermédiaires. Une astuce est de déconstruire les deux côtés de la multiplication en même temps, en utilisant le fait que $(n+1) \times (m+1) = n \times m + n + m + 1$. Ainsi, on définit inductivement la multiplication comme suit :

— Pour tout $b \in \mathcal{P}, \mathbf{mult}(Z, b) = Z$
— Pour tout $a \in \mathcal{P}, \mathbf{mult}(a, Z) = Z$
— Pour tout $a, b \in \mathcal{P}, \mathbf{mult}(S(a), S(b)) = S(\mathbf{add}(\mathbf{mult}(a, b), \mathbf{add}(a, b))$

On peut alors montrer la commutativité et la distributivité sans lemme, en utilisant directement les propriétés de l'addition. Par exemple pour la commutativité, montrons par double induction sur $a, b \in \mathcal{P}$ que $\forall a, b \in \mathcal{P}, \mathbf{mult}(a, b) = \mathbf{mult}(b, a)$. Il y a trois cas :

— $a = Z$. $\mathbf{mult}(Z, b) = Z = \mathbf{mult}(b, Z)$.
— $b = Z$: analogue.
— $a = S(a'), b = S(b')$ avec $a', b' \in \mathcal{P}$.

$$
\begin{aligned}
\mathbf{mult}(a, b) &= \mathbf{mult}(S(a'), S(b')) \\
&= S(\mathbf{add}(\mathbf{mult}(a', b'), \mathbf{add}(a', b')) \\
&= S(\mathbf{add}(\mathbf{mult}(b', a'), \mathbf{add}(a', b')) && \text{par HI, car } (a', b') < (a, b) \\
&= S(\mathbf{add}(\mathbf{mult}(b', a'), \mathbf{add}(b', a')) && \text{par commutativité de l'addition} \\
&= \mathbf{mult}(S(b'), S(a')) && \text{par définition de } \mathbf{mult} \\
&= \mathbf{mult}(b, a)
\end{aligned}
$$

La preuve d'associativité est similaire.

On pouvait aussi définir la multiplication comme suit (plus proche de ce qui a été fait pour l'addition) en traduisant que $(n + 1) \times m = n \times m + m$:

— $\mathbf{mult}(Z, b) = Z$
— $\mathbf{mult}(S(a), b) = \mathbf{add}(\mathbf{mult}(a, b), b)$

Dans ce cas, la preuve nécessite plusieurs lemmes, notamment la distributivité de la multiplication sur l'addition, mais tout se fait par induction simple sur a sans piège.

Exercice 2.

Entiers de Péano (partie 2)

Exercice 3. *Expressions*

Question 1. On définit inductivement la fonction n_{op} :

- Pour $n \in \mathbb{N}$, $n_{op}(\mathbf{Int}(n)) = 0$
- Pour $x \in \mathcal{X}, n_{op}(x) = 0$
- Pour $e_1, e_2 \in \mathcal{E}, n_{op}(e_1 + e_2) = 1 + n_{op}(e_1) + n_{op}(e_2)$
- Pour $e_1, e_2 \in \mathcal{E}, n_{op}(e_1 \times e_2) = 1 + n_{op}(e_1) + n_{op}(e_2)$

Question 2. On définit de même la fonction n_{atom} :

- Pour $n \in \mathbb{N}$, $n_{atom}(\mathbf{Int}(n)) = 1$
- Pour $x \in \mathcal{X}, n_{atom}(x) = 1$
- Pour $e_1, e_2 \in \mathcal{E}, n_{atom}(e_1 + e_2) = n_{atom}(e_1) + n_{atom}(e_2)$
- Pour $e_1, e_2 \in \mathcal{E}, n_{atom}(e_1 \times e_2) = n_{atom}(e_1) + n_{atom}(e_2)$

Question 3. Montrons par induction sur la structure des expressions que pour $e \in \mathcal{E}, n_{\text{op}}(e) + 1 = n_{\text{atom}}(e)$.
Il y a 4 cas :

- Si $e = \mathbf{Int}(n)$ avec $n \in \mathbb{N}$, $n_{op}(e) = 0$ et $n_{atom}(e) = 1$: la propriété est vraie
- Si $e = x \in \mathcal{X}$, $n_{op}(e) = 0$ et $n_{atom}(e) = 1$: la propriété est vraie
- Si $e = e_1 + e_2$ avec $e_1, e_2 \in \mathcal{E}$: par hypothèse d'induction, on a $n_{\text{op}}(e_1) + 1 = n_{\text{atom}}(e_1)$ et $n_{\text{op}}(e_2) + 1 = n_{\text{atom}}(e_2)$. Donc :

$$n_{atom}(e) = n_{atom}(e_1) + n_{atom}(e_2) = n_{\text{op}}(e_1) + 1 + n_{\text{op}}(e_2) + 1 = n_{op}(e) + 1$$

- Si $e = e_1 \times e_2$ avec $e_1, e_2 \in \mathcal{E}$, c'est analogue.

Question 4. Définissons inductivement une fonction $\mathbf{eval}(e, \sigma)$ qui, étant donné une expression e et un contexte σ, calcule la valeur de e avec le contexte σ :

- $\mathbf{eval}(\mathbf{Int}(n), \sigma) = n$
- $\mathbf{eval}(x, \sigma) = \sigma(x)$
- $\mathbf{eval}(e_1 + e_2, \sigma) = \mathbf{eval}(e_1, \sigma) + \mathbf{eval}(e_2, \sigma)$
- $\mathbf{eval}(e_1 \times e_2, \sigma) = \mathbf{eval}(e_1, \sigma) \times \mathbf{eval}(e_2, \sigma)$

Il faut bien faire attention ici que le signe + a deux sens : dans le monde des expressions $\mathcal{E}$, c'est juste une notation syntaxique, et dans le monde des entiers $\mathbb{N}$, c'est l'addition standard. Par exemple, $e_1 + (e_2 + e_3)$ et $(e_1 + e_2) + e_3$ sont deux expressions différentes, mais lorsqu'on les évalue, on obtient le même résultat.

Exercice 4. *Preuve de tri*

Pour rappel, en cours nous avons cherché une version alternative de l'exercice en faisant toutes les fonctions en récursif terminal. La correction proposée ici est pour l'exercice de base.

Question 1. Définition de est_triee :

```
let rec est_triee (l: 'a list) : bool =
  match l with
  | [] | [_] -> true
  | x :: y :: q -> x <= y && est_triee (y :: q)
```

Question 2. Définition de occ :

```
let rec occ (x: 'a) (l: 'a list) : int =
  match l with
  | [] -> 0
  | y :: q -> (if x = y then 1 else 0) + occ x q
```

Dans la suite, on notera $\delta_{x=y} = (1 \text{ si } x = y \text{ et } 0 \text{ sinon})$.

Question 3. Montrons par induction structurelle sur les listes que pour toute liste L, on a $P(L)$: "pour tout élément x, $\mathbf{occ}(x, \mathbf{insert}(x, L)) = \mathbf{occ}(x, L) + 1$". Il y a deux cas :

— $L = []$:

$$\begin{aligned}
\mathbf{occ}(x, \mathbf{insert}(x, L)) &= \mathbf{occ}(x, \mathbf{insert}(x, [])) \\
&= \mathbf{occ}(x, x :: []) \\
&= 1 + \mathbf{occ}(x, []) \\
&= 1 \\
&= \mathbf{occ}(x, []) + 1 \\
&= \mathbf{occ}(x, L) + 1
\end{aligned}$$

Donc $P([])$ est vérifiée.

— $L = y :: Q$ avec y un élément et Q une liste. Hypothèse d'induction : $P(Q)$. On a deux sous-cas à traiter : $x \leq y$ et $x > y$.

— Si $x \leq y$:

$$\begin{aligned}
\mathbf{occ}(x, \mathbf{insert}(x, L)) &= \mathbf{occ}(x, \mathbf{insert}(x, y :: Q)) \\
&= \mathbf{occ}(x, x :: y :: Q) \\
&= 1 + \mathbf{occ}(x, y :: Q) \\
&= 1 + \mathbf{occ}(x, L)
\end{aligned}$$

— Si $x > y$:

$$\begin{aligned}
\mathbf{occ}(x, \mathbf{insert}(x, L)) &= \mathbf{occ}(x, \mathbf{insert}(x, y :: Q)) \\
&= \mathbf{occ}(x, y :: \mathbf{insert}(x, Q)) \\
&= \delta_{x=y} + \mathbf{occ}(x, \mathbf{insert}(x, Q)) \\
&= \delta_{x=y} + 1 + \mathbf{occ}(x, Q) && \text{par HI} \\
&= (\delta_{x=y} + \mathbf{occ}(x, Q)) + 1 \\
&= (\mathbf{occ}(x, y :: Q)) + 1 \\
&= (\mathbf{occ}(x, L)) + 1
\end{aligned}$$

Dans tous les cas, $P(L)$ est vérifiée.

Donc, par principe d'induction structurelle, P est vraie sur toute liste L.

Question 4. Quasiment la même preuve que la question d'avant.

Question 5. Pour le faire totalement proprement, il faut un lemme intermédiaire comme pour la preuve de commutativité de l'addition des entiers de Péano. Commençons par montrer la propriété suivante par induction sur L une liste :

$$\forall L \text{ liste}, P(L) : \text{“}\forall x, y \text{ éléments, si } \mathbf{estTriee}(y :: L) \text{ et } x > y, \text{ alors } \mathbf{estTriee}(y :: \mathbf{insert}(x, L))\text{”}$$

Il y a deux cas à traiter :

- $L = []$. Alors, $\mathbf{insert}(x, L) = x :: []$, donc il faut vérifier que $\mathbf{estTriee}(y :: x :: [])$, ce qui est le cas car $y \leq x$ et $\mathbf{estTriee}(x :: [])$.
- $L = z :: Q$ avec z un élément et Q une liste. On suppose $P(Q)$, montrons $P(L)$. Soient x, y deux éléments, tels que $\mathbf{estTriee}(y :: L)$ et $x > y$. On a donc $\mathbf{estTriee}(y :: z :: Q)$, i.e. $y \leq z$ et $\mathbf{estTriee}(z :: Q)$. Alors, on distingue deux sous-cas :
  - Si $x \leq z$:

$$\begin{aligned} \mathbf{estTriee}(y :: \mathbf{insert}(x, L) &\iff \mathbf{estTriee}(y :: \mathbf{insert}(x, z :: Q) \\ &\iff \mathbf{estTriee}(y :: x :: z :: Q) \\ &\iff y \leq x \text{ et } x \leq z \text{ et } \mathbf{estTriee}(z :: Q) \end{aligned}$$

    Or, les trois conditions sont vérifiées. Donc, $\mathbf{estTriee}(y :: \mathbf{insert}(x, L)$
  - Si $x > z$:

$$\begin{aligned} \mathbf{estTriee}(y :: \mathbf{insert}(x, L) &\iff \mathbf{estTriee}(y :: \mathbf{insert}(x, z :: Q) \\ &\iff \mathbf{estTriee}(y :: z :: \mathbf{insert}(x, Q)) \\ &\iff y \leq z \text{ et } \mathbf{estTriee}(z :: \mathbf{insert}(x, Q)) \end{aligned}$$

    Comme $y \leq z$, il suffit de vérifier la deuxième partie du et.
    Or, $x > z$, et $z :: Q$ est triée, on peut donc appliquer l'hypothèse d'induction, qui donne directement que $\mathbf{estTriee}(z :: \mathbf{insert}(x, Q))$.
    Ce qui permet de conclure ce cas.

Maintenant que l'on dispose de ce lemme, on peut montrer le résultat principal. Montrons par induction structurelle sur les listes que pour toute liste L, pour tout élément x, si L est triée alors $\mathbf{insert}(x, L)$ l'est aussi. Il y a deux cas :

- $L = [] : \forall x, \mathbf{insert}(x, L) = [x]$ est triée.
- $L = y :: Q$ avec y un élément et Q une liste. L'hypothèse d'induction est que pour tout élément x, $\mathbf{estTriee}(Q) \Rightarrow \mathbf{estTriee}(\mathbf{insert}(x, Q))$. Supposons que $\mathbf{estTriee}(L)$. On distingue à nouveau deux cas :
  - Si $x \leq y$:

$$\begin{aligned} \mathbf{estTriee}(\mathbf{insert}(x, L)) &\iff \mathbf{estTriee}(\mathbf{insert}(x, y :: Q)) \\ &\iff \mathbf{estTriee}(x :: y :: Q) \\ &\iff x \leq y \text{ et } \mathbf{estTriee}(y :: Q) \\ &\iff x \leq y \text{ et } \mathbf{estTriee}(L) \end{aligned}$$

    Or, les deux conditions sont réalisées. Donc, $\mathbf{estTriee}(\mathbf{insert}(x, L))$.

— Si $x > y$: **estTriee**$(y :: Q)$, donc le lemme nous permet d'affirmer que **estTriee**$(y ::$ **insert**$(x, Q))$. De plus on a :

$$\begin{aligned} \textbf{estTriee}(\textbf{insert}(x, L)) &\iff \textbf{estTriee}(\textbf{insert}(x, y :: Q)) \\ &\iff \textbf{estTriee}(y :: \textbf{insert}(x, Q)) \end{aligned}$$

Ce qui conclut ce cas d'induction.

Question 6. Pas de difficulté pour cette question, on procède par induction structurelle et l'étape d'induction est presque immédiate, en utilisant le résultat précédent.

Question 7. Commençons par déterminer la complexité de `insert`. On note $C(n)$ le coût de `insert` sur une liste de taille n. On a $C(0) = A$ une certaine constante et $C(n) \leq C(n-1) + \mathcal{O}(1)$. Donc $C(n) = \mathcal{O}(n)$. En effet, en notant $n_0 \in \mathbb{N}$ et $K_1 > 0$ tels que $\forall n \geq n_0, C(n) \leq C(n-1) + K_1$:

$$\begin{aligned} C(n) &\leq C(n-1) + K_1 \\ &\leq C(n-2) + 2K_1 \\ &\leq \dots \\ &\leq C(n-k) + kK_1 \end{aligned}$$

pour $k \leq n - n_0$. En particulier pour $k = n - n_0, C(n) \leq C(n_0) + (n - n_0)K_1 = \mathcal{O}(n)$.

Ensuite, en notant $D(n)$ le coût de `insert_sort` sur une liste de taille n, on a $D(0) = B$ une certaine constante, et $D(n) \leq D(n-1) + \mathcal{O}(n)$. Donc, $D(n) = \mathcal{O}(n^2)$ (la preuve est analogue).

Question 8. On commence par définir la fonction `fusion` :

```
1 let rec fusion (l1: 'a list) (l2: 'a list) : 'a list =
2   match (l1, l2) with
3   | [], _ -> l2
4   | _, [] -> l1
5   | x1::q1, x2::q2 ->
6       if x1 <= x2 then x1 :: fusion q1 l2
7         else           x2 :: fusion l1 q2
```

Montrons par induction double sur (L_1, L_2) un couple de liste que

$$\forall L_1, L_2 \text{ listes, } P(L_1, L_2) : \text{“}\forall x, \textbf{occ}(x, \textbf{fusion}(L_1, L_2)) = \textbf{occ}(x, L_1) + \textbf{occ}(x, L_2)\text{”}$$

On distingue 3 cas :

— $L_1 = []$: Soit x un élément. **fusion**$(L_1, L_2) = L_2$ et **occ**$(x, L_1) = 0$, donc on a bien **occ**$(x,$ **fusion**$(L_1, L_2)) =$ **occ**$(x, L_1) +$ **occ**(x, L_2).

— $L_2 = []$: analogue

— $L_1 = x_1 :: Q_1, L_2 = x_2 :: Q_2$ avec x_1, x_2 des éléments et Q_1, Q_2 des listes. Hypothèses d'induction : $P(Q_1, L_2)$ et $P(L_1, Q_2)$. Si $x_1 \leq x_2$:

$$\begin{aligned} \textbf{occ}(x, \textbf{fusion}(L_1, L_2)) &= \textbf{occ}(x, x_1 :: \textbf{fusion}(Q_1, L_2)) \\ &= \delta_{x=x_1} + \textbf{occ}(x, \textbf{fusion}(Q_1, L_2)) \\ &= \delta_{x=x_1} + \textbf{occ}(x, Q_1) + \textbf{occ}(x, L_2)) \quad \text{par HI} \\ &= \textbf{occ}(x, L_1) + \textbf{occ}(x, L_2) \end{aligned}$$

Le reste des preuves sur **fusion** se fait selon le même schéma. Il faut ensuite refaire le même travail sur la fonction **separer** qui permet de couper une liste en deux, puis enfin utiliser les résultats prouvés pour montrer la correction du tri fusion.

Exercice 5.

Preuves sur les listes

Exercice 6.

Induction bien fondée

Exercice 7.

Mi casa es tu casa

Exercice 8.

Le retour de Knaster-Tarski

Question 1. Montrons que toute fonction continue est croissante. Soit $f : (X, \leq_X) \to (Y, \leq_Y)$ continue, avec X, Y des treillis complets. Soient $x_1, x_2 \in X$ avec $x_1 \leq x_2$. On considère $A = \{x_1, x_2\}$. A est non vide et totalement ordonnée, donc $f(\bigvee A) = \bigvee(f(A))$. De plus, $\bigvee(A) = x_2$. Donc, $f(x_2) = \bigvee(\{f(x_1), f(x_2)\})$, et donc $f(x_2) \geq f(x_1)$. La fonction f est bien croissante.

Question 2. $\bot$ est le plus petit élément de X. On pose $p_0 = \bot$, et $p_{n+1} = f(p_n)$ pour $n \in \mathbb{N}$. Montrons que la suite $(p_n)_n$ est croissante. $p_0 \leq p_1$ car $p_0 = \bot$. Donc, pour $n \in \mathbb{N}$, en appliquant f n fois, on obtient par croissance de f que $f^n(p_0) \leq f^{n+1}(p_1)$, i.e. que $p_n \leq p_{n+1}$. La suite est bien croissante.

Question 3. Montrons que $p = \bigvee\{p_n | n \in \mathbb{N}\}$ est un point fixe de f. Par continuité de f, $f(p) = f(\bigvee\{p_n | n \in \mathbb{N}\}) = \bigvee\{f(p_n) | n \in \mathbb{N}\}$. Donc, $f(p) = \bigvee\{p_{n+1} | n \in \mathbb{N}\}$, et comme $(p_n)_n$ est croissante, cela donne bien $f(p) = \bigvee\{p_n | n \in \mathbb{N}\} = p$.

Montrons maintenant que c'est le plus petit point fixe. Soit p' un autre point fixe de f. $p' \geq \bot = p_0$, donc par croissance de f, en itérant f sur l'inégalité, on obtient que $f^n(p') \geq p_n$ pour $n \in \mathbb{N}$. Donc, $p' \geq p_n$ pour $n \in \mathbb{N}$. Donc p' est un majorant de $\{p_n | n \in \mathbb{N}\}$, donc $p' \geq p$.

Question 4. $\Phi(\emptyset)$ vaut $\emptyset \cup B \cup \emptyset = B$: c'est l'ensemble des termes construits avec les règles de base. $\Phi(\Phi(\emptyset))$ contient B ainsi que l'ensemble des termes construits à partir des règles d'induction sur les éléments de B, autrement dit les termes de hauteur 1, autrement dit ce que l'on a noté X_1 dans le cours. De manière générale, $\Phi^n(\emptyset)$ représente les termes inductifs de hauteur au plus $n - 1$ pour $n \in \mathbb{N}^*$.

Question 5. Commençons par montrer la croissante de Φ. Si $A \subseteq A'$ sont deux parties de X, alors $f(A) \subseteq f(A')$. En effet :

- $A \subseteq A'$
- $B \subseteq B$
- en notant $I_A = \{S(p, x_1, \ldots, x_r) \mid (S, r, P) \in \mathcal{I}, x_1, \ldots, x_r \in A, p \in P\}$, on a bien $I_A \subseteq I'_A$: si $S(p, x_1, \ldots, x_r) \in I_A$, alors $x_1, \ldots, x_r \in A$ et donc $x_1, \ldots x_r \in A'$, et donc $S(p, x_1, \ldots, x_r) \in I'_A$).

Continuité : pas beaucoup plus compliqué, il faut juste dérouler les définition.

Question 6. Le plus petit point fixe de Φ est donc la limite des $\Phi^n(\emptyset)$, autrement dit c'est $\bigcup_{n\in\mathbb{N}} X_n$, c'est à dire X tout entier.

Question 7. Montrons que $\Phi(Q) = Q$. Cela permettra de montrer que Q est un point fixe, et donc que $Q = X$ d'après la question précédente.

- $f(Q)$ contient Q par définition de Φ.
- Q contient $\Phi(Q)$ car Q contient B, Q et I_Q : en effet si $x = S(p, x_1, \ldots, x_r) \in I_Q$ avec $x_1, \ldots, x_r \in Q$, alors $x_1, \ldots, x_r$ vérifient $\mathcal{Q}$, et donc x aussi par hypothèse sur $\mathcal{Q}$.

D'où l'égalité.

Donc, $Q = X$, autrement dit tous les éléments de X vérifient Q.