Exercice 4 (Arithmétique de Peano en Coq)

En `Coq`, la théorie de l'arithmétique de Peano peut être implémentée comme suit :

```
Section Peano.

Parameter N : Set.
Parameter o : N.
Parameter s : N -> N.
Parameters plus mult : N -> N -> N.
Variables x y : N.
Axiom ax1 : ~((s z) = o).
Axiom ax2 : exists z, ~(x = o) -> (s z) = x.
Axiom ax3 : (s x) = (s y) -> x = y.
Axiom ax4 : (plus x o) = x.
Axiom ax5 : (plus x (s y)) = (s (plus x y)).
Axiom ax6 : (mult x o) = o.
Axiom ax7 : (mult x (s y)) = (plus (mult x y) x).

End Peano.
```

1. Démontrer les propositions suivantes dans cette théorie en `Coq` :
   - $1 + 2 = 3$
   - $2 + 2 = 4$
   - $2 \times 2 = 4$
2. Écrire une tactique qui calcule automatiquement dans cette théorie.
3. Démontrer les propositions précédentes à l'aide de cette tactique.
4. Même question en utilisant la tactique `autorewrite` (voir la documentation).

▶ Voir TP2

3 TD/TP3 : Types inductifs

Exercice 1 (Fonctions et preuves inductives sur les entiers)

1. Écrire la fonction *mult* sur les entiers naturels $\mathcal{N}$.
2. Démontrer que : $\forall n \in \mathcal{N}.mult(2, n) = plus(n, n)$.
3. Démontrer que : $\forall n \in \mathcal{N}.mult(n, 2) = plus(n, n)$.

1. Soit $mult : \mathcal{N} \times \mathcal{N} \to \mathcal{N}$.
   (m_1) $\forall n \in \mathcal{N}.mult(0, n) = 0$.
   (m_2) $\forall p, n \in \mathcal{N}.mult((S\ p), n) = plus(mult(p, n), n)$.

2. On sait que $2 = (S\ (S\ 0))$. Ainsi :

$$\begin{aligned} mult((S\ (S\ 0)), n) &= plus(mult((S\ 0), n), n) && \text{par } m_2 \\ &= plus(plus(mult(0, n), n), n) && \text{par } m_2 \\ &= plus(plus(0, n), n) && \text{par } m_1 \\ &= plus(n, n) && \text{par } plus(0, n) = n \end{aligned}$$

3. On rappelle la définition de $plus$:

   (p_1) $\forall n \in \mathcal{N}.plus(0, n) = n$

   (p_2) $\forall p, n \in \mathcal{N}.plus((S\ p), n) = (S\ plus(p, n))$

   Lemme 1. $\forall n \in \mathcal{N}.plus(n, 0) = n$

   Démonstration. On prouve le lemme 1 par induction structurelle :

   Base Pour $n = 0$, on a $plus(0, 0) = 0$ par p_1. Par réflexivité, la propriété est vérifiée pour le cas de base.

   Induction On suppose que $plus(n, 0) = n$. Montrons que $plus((S\ n), 0) = (S\ n)$.

$$\begin{aligned} plus((S\ n), 0) &= (S\ plus(n, 0)) && \text{par } p_2 \\ &= (S\ n) && \text{par hypothèse d'induction} \end{aligned}$$

   Par réflexivité, la propriété est vérifiée $\forall n \in \mathcal{N}$.

   □

   Lemme 2. $\forall n, m \in \mathcal{N}.plus(m, (S\ n)) = (S\ plus(m, n))$

   Démonstration. On prouve le lemme 2 par induction structurelle :

   Base Pour $m = 0$, on a $plus(0, (S\ n)) = (S\ n)$ par p_1, et $(S\ plus(0, n)) = (S\ n)$. Par réflexivité, $(S\ n) = (S\ n)$ et donc la propriété est vérifiée pour le cas de base.

   Induction On suppose que la propriété est vraie pour un m quelconque, c'est à dire que $plus(m, (S\ n)) = (S\ plus(m, n))$. Montrons que $plus((S\ m), (S\ n)) = (S\ (S\ plus(m, n)))$.

$$\begin{aligned} plus((S\ m), (S\ n)) &= (S\ plus(m, (S\ n))) && \text{par } p_2 \\ &= (S\ (S\ plus(m, n)) && \text{par hypothèse d'induction} \end{aligned}$$

   Par réflexivité, on a bien $plus((S\ m), (S\ n)) = (S\ (S\ plus(m, n)))$, donc la propriété est vérifiée $\forall n, m \in \mathcal{N}$.

   □

   On peut maintenant montrer $\forall n \in \mathcal{N}.mult(n, 2) = plus(n, n)$ par induction :

   Base Pour $n = 0$, on a $mult(0, 2) = 0$ par m_1, et $plus(0, 0) = 0$ par p_1. Par réflexivité, $mult(0, 2) = plus(0, 0)$ et ainsi la propriété est vérifiée pour le cas de base.

   Induction On suppose que $\forall n \in \mathcal{N}.mult(n, 2) = plus(n, n)$. Montrons que $\forall n \in \mathcal{N}.mult((S\ n), 2) = plus((S\ n), (S\ n))$. On sait que $2 = (S\ (S\ 0))$.

$$\begin{aligned} mult((S\ n), (S\ (S\ 0))) &= plus(mult(n, 2), (S\ (S\ 0))) && \text{par } m_2 \\ &= plus(plus(n, n), (S\ (S\ 0))) && \text{par hypothèse d'induction} \\ &= (S\ plus(plus(n, n), (S\ 0))) && \text{par le lemme 2} \\ &= (S\ (S\ plus(plus(n, n), 0))) && \text{par le lemme 2} \\ &= (S\ (S\ plus(n, n))) && \text{par le lemme 1} \end{aligned}$$

De plus :

$$\begin{aligned} plus((S\ n),(S\ n)) &= (S\ plus(n,(S\ n))) && \text{par } p_2 \\ &= (S\ (S\ plus(n,n))) && \text{par le lemme 2} \end{aligned}$$

Par réflexivité, on conclut que $mult((S\ n),2) = plus((S\ n),(S\ n))$ et donc la propriété est vérifiée $\forall n \in \mathcal{N}$.

Exercice 2 (Fonctions et preuves inductives sur les listes)

1. Écrire la fonction rev qui inverse les éléments d'une liste.
2. Démontrer que : $\forall l \in \mathcal{L}, \forall e \in \mathcal{A}. rev(app(l,[e])) = e :: rev(l)$.
3. Démontrer que : $\forall l \in \mathcal{L}. rev(rev(l)) = l$.

1. On rappelle que l'ensemble $\mathcal{A}$ est le type des éléments de la liste. On peut définir $rev : \mathcal{L} \to \mathcal{L}$:
   (l_1) $rev([\,]) = [\,]$
   (l_2) $\forall e \in \mathcal{A}, \forall l \in \mathcal{L}. rev(e :: l) = app(rev(l),[e])$.
2. On rappelle la définition de app :
   (a_1) $\forall l \in \mathcal{L}. app([\,],l) = l$
   (a_2) $\forall e \in \mathcal{A}, \forall l_1, l_2 \in \mathcal{L}. app(e :: l_1, l_2) = e :: app(l_1, l_2)$.

   Lemme 3. $\forall l \in \mathcal{L}, \forall e \in \mathcal{A}. rev(app(l,[e])) = e :: rev(l)$

   Démonstration. On le démontre par induction.

   Base Pour $l = [\,]$ et $e \in \mathcal{A}$, on a $rev(app([\,],[e])) = rev([e]) = [e]$ et $e :: rev([\,]) = e :: [\,] = [e]$. Par réflexivité, $rev(app([\,],e)) = e :: rev([\,])$.

   Induction On suppose que $\forall l \in \mathcal{L}, \forall e_1 \in \mathcal{A}. rev(app(l,[e_1])) = e_1 :: rev(l)$. Montrons que $\forall l \in \mathcal{L}, \forall e_1, e_2 \in \mathcal{A}. rev(app(e_2 :: l,[e_1])) = e_1 :: rev(e_2 :: l)$.

$$\begin{aligned} rev(app(e_2 :: l,[e_1])) &= rev(e_2 :: app(l,[e_1])) && \text{par } a_2 \\ &= app(rev(app(l,[e_1])),[e_2]) && \text{par } l_2 \\ &= app(e_1 :: rev(l),[e_2]) && \text{par hypothèse d'induction} \\ &= e_1 :: app(rev(l),[e_2]) && \text{par } a_2 \\ &= e_1 :: rev(e_2 :: l) && \text{par } l_2 \end{aligned}$$

   Par réflexivité, $rev(app(e_2 :: l,[e_1])) = e_1 :: rev(e_2 :: l)$, donc la propriété est vérifiée $\forall e \in \mathcal{A}$ et $\forall l \in \mathcal{L}$.

   □

3. On démontre $\forall l \in \mathcal{L}. rev(rev(l)) = l$ par induction.

   Base Pour $l = [\,]$, on a $rev(rev([\,])) = rev([\,]) = [\,]$. Par réflexivité, la propriété est vérifiée pour le cas de base.

   Induction On suppose que $\forall l \in \mathcal{L}. rev(rev(l)) = l$. Montrons que $\forall e \in \mathcal{A}, \forall l \in \mathcal{L}. rev(rev(e :: l)) = e :: l$.

$$\begin{aligned} rev(rev(e :: l)) &= rev(app(rev(l),[e])) && \text{par } l_2 \\ &= e :: rev(rev(l)) && \text{par le lemme 3} \\ &= e :: l && \text{par hypothèse d'induction} \end{aligned}$$

   Par réflexivité, $rev(rev(e :: l)) = e :: l$, donc la propriété est vérifiée $\forall l \in \mathcal{L}$.

Exercice 3 (Type inductif des formules en logique)

1. Définir le type des formules en logique propositionnelle.
2. Écrire la fonction *sub*, qui rend l'ensemble des sous-formules d'une formule F.
3. Écrire la fonction *nbc*, qui rend l'ensemble des connecteurs d'une formule F.
4. Écrire le schéma d'induction structurelle des formules.
5. Démontrer que : $|sub(F)| \leq 2 \times nbc(F) + 1$ pour toute formule F.

1. Soit $\mathcal{V}$ l'ensemble des variables de propositions. Soit $\mathcal{F}$ l'ensemble des formules de logique propositionnelle. On a $\mathcal{F}$:
   - Si $A \in \mathcal{V}$, alors $A \in \mathcal{F}$,
   - $\top, \bot \in \mathcal{F}$,
   - $\forall \phi \in \mathcal{F}, \neg\phi \in \mathcal{F}$,
   - $\forall \phi_1, \phi_2 \in \mathcal{F}, \phi_1\ c\ \phi_2 \in \mathcal{F}$ (avec $c \in \{\vee, \wedge, \Rightarrow, \Leftrightarrow\}$)
2. $sub : \mathcal{F} \to 2^{\mathcal{F}}$:
   - $sub(\top) = \{\top\}$; $sub(\bot) = \{\bot\}$,
   - $\forall A \in \mathcal{V}.sub(A) = \{A\}$,
   - $\forall \phi \in \mathcal{F}.sub(\neg\phi) = \{\neg\phi\} \cup sub(\phi)$,
   - $\forall \phi_1, \phi_2 \in \mathcal{F}.sub(\phi\ c\ \phi_2) = \{\phi_1\ c\ \phi_2\} \cup sub(\phi_1) \cup sub(\phi_2)$ (avec $c \in \{\vee, \wedge, \Rightarrow, \Leftrightarrow\}$).
3. $nbc : \mathcal{F} \to \mathcal{N}$:
   - $nbc(\top) = 0$; $nbc(\bot) = 0$,
   - $\forall A \in \mathcal{V}.nbc(A) = 0$,
   - $\forall \phi \in \mathcal{F}.nbc(\neg\phi) = 1 + nbc(\phi)$,
   - $\forall \phi_1, \phi_2 \in \mathcal{F}.nbc(\phi\ c\ \phi_2) = 1 + nbc(\phi_1) + nbc(\phi_2)$ (avec $c \in \{\vee, \wedge, \Rightarrow, \Leftrightarrow\}$).
4. $\forall P \in (\mathcal{F} \to Prop).(\forall A \in \mathcal{V}.P(A)) \Rightarrow (\forall \phi \in \mathcal{F}.P(\phi) \Rightarrow P(\neg\phi)) \Rightarrow (\forall \phi_1, \phi_2 \in \mathcal{F}.(P(\phi_1) \wedge P(\phi_2) \Rightarrow P(\phi_1 \wedge \phi_2) \wedge (P(\phi_1) \vee P(\phi_2) \Rightarrow P(\phi_1 \vee \phi_2)) \wedge (P(\phi_1) \Rightarrow P(\phi_2) \Rightarrow P(\phi_1 \Rightarrow \phi_2)) \wedge (P(\phi_1) \Leftrightarrow P(\phi_2) \Rightarrow P(\phi_1 \Leftrightarrow \phi_2)))) \Rightarrow \forall \phi \in \mathcal{F}.P(\phi)$
5. On démontre $\forall F \in \mathcal{F}.|sub(F)| \leq 2 \times nbc(F) + 1$ par induction :

   Base $F \in \mathcal{V}$ ou $F \in \{\top, \bot\}$, alors $sub(F) = \{F\}$ et $nbc(F) = 2 \times 0 + 1 = 1$. Par réflexivité, $|\{F\}| = 1$, donc la propriété est vérifiée pour le cas de base.

   Induction On suppose que $\forall \phi, \phi_1, \phi_2 \in \mathcal{F}.|sub(\phi)| \leq 2 \times nbc(\phi) + 1$.
   - $F = \neg\phi$, dans ce cas, $sub(F) = \{F\} \cup sub(\phi)$. On a $|sub(F)| = 1 + |sub(\phi)| \leq 1 + 2 \times nbc(\phi) + 1 = 2nbc(\phi) + 2$ et $nbc(F) = 1 + nbc(\phi)$, on a $2(nbc(F)) + 1 = 2nbc(\phi) + 2$, on a donc bien $|sub(F)| \leq 2(nbc(F)) + 1$.
   - $F = \phi_1\ c\ \phi_2$ avec $c \in \{\vee, \wedge, \Rightarrow, \Leftrightarrow\}$. Dans ce cas, $sub(F) = \{F\} \cup sub(\phi_1) \cup sub(\phi_2)$. On a $|sub(F)| = 1 + |sub(\phi_1)| + |sub(\phi_2)| \leq 1 + 2nbc(\phi_1) + 1 + 2nbc(\phi_2) + 1 = 2(nbc(\phi_1) + nbc(\phi_2)) + 3$ et $nbc(F) = 1 + nbc(\phi_1) + nbc(\phi_2)$, on a $2(nbc(F)) + 1 = 2(nbc(\phi_1) + nbc(\phi_2)) + 3$. On a donc bien $|sub(F)| \leq 2(nbc(F)) + 1$.

Exercice 4 (Relations inductives sur les listes)

1. Spécifier la relation « être une permutation de » pour deux listes.
2. Démontrer que la liste $[1; 2; 3]$ est une permutation de $[3; 2; 1]$.
3. Spécifier la relation « être triée » pour une liste.
4. Démontrer que la liste $[1; 2; 3]$ est triée.

1. On défini la relation inductive $is\_perm : \mathcal{L} \times \mathcal{L} \to Prop$ de la façon suivante :
   ($perm_1$) On a : $is\_perm([\,], [\,])$,
   ($perm_2$) On a aussi : $\forall e \in \mathcal{A}.is\_perm([e], [e])$.
   ($perm_3$) $\forall l_1, l_2, l_3 \in \mathcal{L}.is\_perm(l_1, app(l_2, l_3)) \Rightarrow is\_perm(e :: l_1, app(l_2, e :: l_3))$.
2. On veut montrer que $is\_perm([1; 2; 3], [3; 2; 1])$.
   — Par $perm_3$, on veut montrer $is\_perm([2; 3], [3; 2])$.
   — Par $perm_3$, on veut montrer que $is\_perm([2], [2])$.
   — Démontré en utilisant le cas de base.
3. On défini la relation inductive $is\_sorted : \mathcal{L} \to Prop$ de la façon suivante :
   ($sort_1$) On a $is\_sorted([\,])$,
   ($sort_2$) On a $\forall n \in \mathcal{N}.is\_sorted([n])$,
   ($sort_3$) $\forall e, n \in \mathcal{N}.\forall l \in \mathcal{L}.is\_sorted(l + +[e]) \wedge e \leq n \Rightarrow is\_sorted(l + +[e; n])$.
4. On veut montrer que $is\_sorted([1; 2; 3])$.
   — Par $sort_3$, comme $2 < 3$, on veut montrer $is\_sorted([1; 2])$.
   — Par $sort_3$, comme $1 < 2$, on veut montrer $is\_sorted([1])$.
   — Démontré en utilisant $sort_2$.

Exercice 5 (Preuves en Coq)

Faire les exercices 1, 2 et 3 en Coq.

▶ Voir TP3

Exercice 6 (Preuves en Coq)

1. Écrire la relation inductive is_even (vue en cours).
2. Écrire une tactique qui démontre des buts de la forme $is\_even(n)$.
3. Écrire une tactique qui démontre les buts de la forme $\neg is\_even(n)$.
4. Écrire une tactique qui démontre les buts précédents indifféremment.
5. Écrire la fonction f_{is_even} qui teste si un entier est pair.
6. Démontrer que la fonction f_{is_even} est correcte vis à vis de la relation is_even.

▶ Voir TP3

4 TD/TP4 : Preuves par induction

Exercice 1 (Fonction factorielle)

1. Spécifier la fonction factorielle à l'aide d'une relation inductive.
2. Écrire la fonction factorielle.
3. Écrire le schéma d'induction fonctionnelle associé à cette fonction.
4. Démontrer la correction de la fonction en utilisant le schéma d'induction structurel.
5. Démontrer la correction de la fonction en utilisant le schéma d'induction fonctionnel.
6. Démontrer la complétude de la fonction en utilisant le schéma d'induction sur la relation.
7. Répondre aux questions précédentes en utilisant Coq.

1. Soit la relation inductive $is\_fact : \mathbb{N} \times \mathbb{N} \to Prop$:
   $(fact_1)$ On a : $is\_fact(0, 1)$;
   $(fact_2)$ $\forall n, p \in \mathbb{N}. is\_fact(n, p) \Rightarrow is\_fact((S\ n), p \times (S\ n))$.
2. On peut définir $fact : \mathbb{N} \to \mathbb{N}$:
   (f_1) $fact(0) = 1$
   (f_2) $\forall p \in \mathbb{N}. fact((S\ p)) = (S\ p) \times fact(p)$
3. $\forall P \in \mathbb{N} \times \mathbb{N} \to Prop. P(0, 1) \Rightarrow (\forall n \in \mathbb{N}. P(n, fact(n)) \Rightarrow P((S\ n), (S\ n) \times fact(n))) \Rightarrow \forall n \in \mathbb{N}. P(n, fact(n))$
4. On veut montrer que $\forall n, p \in \mathbb{N}. fact(n) = p \Rightarrow is\_fact(n, p)$ par induction structurelle :
   Base Soit $n = 0$. $\forall p \in \mathbb{N}. fact(0) = p \Rightarrow is\_fact(0, p)$. Or, $fact(0) = 1$, donc on veut montrer que $\forall p \in \mathbb{N}. 1 = p \Rightarrow is\_fact(0, p)$. On peut donc remplacer p par 1, et on doit démontrer que $is\_fact(0, 1)$, ce qui est le cas de base de la spécification inductive de is_fact.
   Induction On suppose que $\forall n, p \in \mathbb{N}. fact(n) = p \Rightarrow is\_fact(n, p)$. On veut montrer que $\forall n, p \in \mathbb{N}. fact((S\ n)) = (S\ n) \times p \Rightarrow is\_fact((S\ n), (S\ n) \times p)$:
   $$\begin{aligned} is\_fact((S\ n), (S\ n) \times p) &= is\_fact((S\ n), fact((S\ n))) && \text{par } fact((S\ n)) = (S\ n) \times p \\ &= is\_fact((S\ n), fact(n) \times (S\ n)) && \text{par la définition de } fact \\ &= is\_fact(n, fact(n)) && \text{par } fact_2 \\ &= fact(n) = fact(n) && \text{par } p = fact(n) \end{aligned}$$
   Par réflexivité, on a $is\_fact((S\ n), (S\ n) \times p) = \top$ si et seulement si $is\_fact(n, fact(n))$.
5. On veut montrer que $\forall n, p \in \mathbb{N}. fact(n) = p \Rightarrow is\_fact(n, p)$ par induction fonctionnelle :
   Base On doit montrer que $fact(0) = 1 \Rightarrow is\_fact(0, 1)$. Par f_1, $1 = 1 \Rightarrow is\_fact(0, 1)$, qui devient $\top \Rightarrow \top$ par $fact_1$. Le cas de base est donc vérifié.
   Induction On suppose $\forall n, p \in \mathbb{N}. fact(n) = p \Rightarrow is\_fact(n, p)$, et que $fact(n) \times (S\ n) = p$, et on veut montrer que $is\_fact((S\ n), p)$:
   $$\begin{aligned} is\_fact((S\ n), p) &= is\_fact((S\ n), fact(n) \times (S\ n)) && \text{par } p = fact(n) \times (S\ n) \\ &= is\_fact(n, fact(n)) && \text{par } fact_2 \\ &= fact(n) = fact(n) && \text{par l'hypothèse d'induction} \end{aligned}$$
   Par réflexivité, on a vérifié la propriété pour tout n.
6. On veut montrer que $\forall n, p \in \mathbb{N}. is\_fact(n, p) \Rightarrow fact(n) = p$ par induction sur la relation :

Base On doit montrer que $is\_fact(0,1) \Rightarrow fact(0) = 1$. Par f_1, on a $1 = 1$, et par réflexivité, le cas de base est vérifié.

Induction On suppose que $\forall n, m \in \mathbb{N}. is\_fact(n,m) \Rightarrow fact(n) = m$ et on veut montrer que $fact((S\ n)) = m \times (S\ n)$:

$$\begin{aligned} fact((S\ n)) &= fact(n) \times (S\ n) && \text{par } f_2 \\ &= m \times (S\ n) && \text{par l'hypothèse d'induction} \end{aligned}$$

On a bien $fact((S\ n)) = m \times (S\ n)$ et donc par réflexivité la propriété est vérifiée pour tout n.

7. ▶ Voir TP4

Exercice 2 (Fonction de parité)

Cet exercice est à faire entièrement en Coq

1. Écrire la relation inductive is_even vue en cours.
2. Écrire la fonction récursive f_{is_even} vue en cours.
3. Démontrer que : $\forall n \in \mathbb{N}. f_{is\_even}(n) = \top \Rightarrow is\_even(n)$.
4. Démontrer que : $\forall n \in \mathbb{N}. f_{is\_even}(n) = \bot \Rightarrow \neg is\_even(n)$.
5. Démontrer que : $\forall n \in \mathbb{N}. is\_even(n) \Rightarrow f_{is\_even}(n) = \top$.
6. Démontrer que : $\forall n \in \mathbb{N}. \neg is\_even(n) \Rightarrow f_{is\_even}(n) = \bot$.

▶ Voir TP4

Exercice 3 (Fonction de pgcd)

Cet exercice est à faire entièrement en Coq

1. Écrire la fonction gcd vue en cours.
2. Définir $divides(r,(a,b))$ qui exprime r divise a et b avec $r \in \mathbb{N}^*$ et $a, b \in \mathbb{N}$.
3. Démontrer que : $\forall a, b, r \in \mathbb{N}^*. gcd(a,b) = r \Rightarrow divides(r,(a,b))$.
4. Définir $bezout(r,(a,b))$ qui exprime qu'il existe $p, q \in \mathbb{Z}$ t.q. $p \times a + q \times b = r, r, a, b \in \mathbb{N}$.
5. Démontrer que : $\forall a, b, r \in \mathbb{N}^*. gcd(a,b) = r \Rightarrow bezout(r,(a,b))$.

▶ Voir TP4

5 TD/TP5 : Preuve de correction du tri par insertion

▶ Voir TP5